This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04N 5/00 — A1

(11) International Publication Number: WO 00/24192

(43) International Publication Date: 27 April 2000 (27.04.00)

(21) International Application Number: PCT/US99/21983

(22) International Filing Date: 22 September 1999 (22.09.99)

(30) Priority Data:
60/104,777 19 October 1998 (19.10.98) US

(71) Applicant *(for all designated States except US)*: GENERAL INSTRUMENT CORPORATION [US/US]; 101 Tournament Drive, Horsham, PA 19044 (US).

(72) Inventor; and

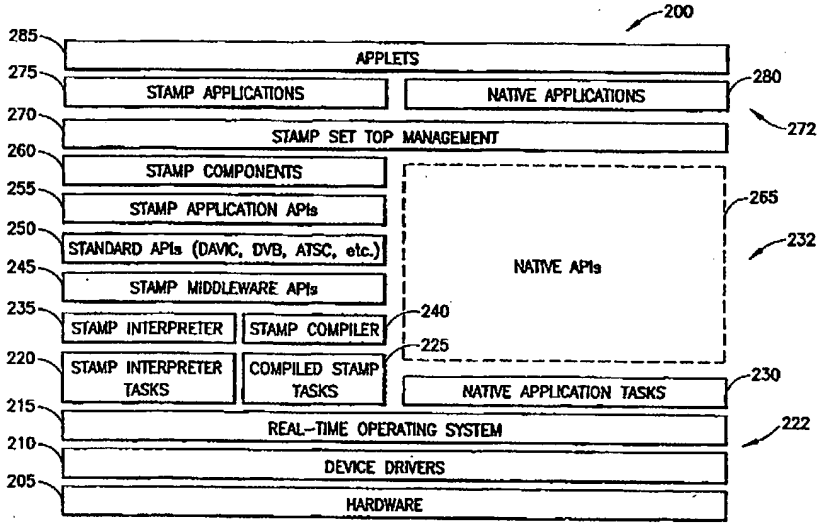
(75) Inventor/Applicant *(for US only)*: MEANDZIJA, Branislav, N. [US/US]; 827 Coast Boulevard South, LaJolla, CA 92037 (US).

(74) Agent: LIPSITZ, Barry, R.; Building No. 8, 755 Main Street, Monroe, CT 06468 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: TELEVISION SET-TOP BOX WITH CONFIGURABLE FUNCTIONALITY



(57) Abstract

A software architecture framework for the soft set top box. A Set Top software architecture (200, 1000) allows the consideration of Set Top application and middleware software functionality (120) independent from the operating system (110) and hardware (100). Various types of functionality can be supported, such as multiple users, secure controlled access of resources, application download, registration, start, stop, and monitoring, resource download, registration, start, stop, and monitoring, and management of audio, video, and data presentations.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

TELEVISION SET-TOP BOX WITH CONFIGURABLE FUNCTIONALITY

BACKGROUND OF THE INVENTION

This application claims the benefit of U.S. Provisional Application No. 60/104,777, filed October 19, 1998.

The present invention provides a software architecture framework for the soft set top and associates the different architectural components with the Opencable Advanced Set Top API Classification.

The current OpenCable API classification is summarized in section 7 of the detailed description, below.

The following acronyms and terms are used:

API - Application Program Interface;
ATSC - Advanced Television Systems Committee;
ATM - Asynchronous Transfer Mode;
CA - Conditional Access;
CORBA - Common Object Request Broker Architecture;
CSLIP - Connectionless Serial Line Interface Protocol;
DASE - ATSC T3/S17 Digital TV Application Software Environment;
DAVIC - Digital Audio-Visual Council;
DC-2 - Digicipher II(tm), digital video standard proprietary to General Instrument (GI) Corporation;
DSMCC - Digital Storage Media Command and Control;

DTV - Digital Television;
DVB - Digital Video Broadcast;
DVD - Digital Video Disc;
EPG - Electronic Program Guide;
FTP - File Transfer Protocol;
GUI - Graphical User Interface;
HSSIO - High-Speed Serial I/O
I/O - Input/Output;
IIOP - Internet Inter Orb Protocol;
IP - Internet Protocol;
IPG - Interactive Program Guide;
IRD - Integrated Receiver Decoder;
ISO - International Standards Organization;
JVM - Java Virtual Machine;
MIB - Management Information Base;
MPEG - Moving Picture Experts Group;
PSIP - Program and System Information Protocol (for Terrestrial Broadcast and Cable);
RAM - Random Access Memory;
RMI - Remote Method Invocation;
Sat - Satellite;
SI - (DVB) Service Information;
SMPTE - Society of Motion Picture and Television Engineers;
SNMP - Simple Network Management Protocol;
SODA - denotes a security chip proprietary to General Instrument Corporation;
S/PDIG -
STAMP - Set Top Applications and Middleware Platform;
S-vid - Video connector interface;

TCP - Transmission Control Protocol;
UHF - Ultra High Frequency;
UML - Unified Modeling Language;
v.34 - a physical interface standard; and
xDSL - high-speed Digital Subscriber Loop modem technology.

A Set Top box, also referred to as an Integrated Received Decoder (IRD) or a subscriber terminal, is a device that receives and decodes television signals for presentation by a television. The signals can be delivered over a satellite, through a cable plant, or by means of terrestrial broadcast. Modern set tops also support video on demand (VOD), pay-per-view, interactive shopping, electronic commerce, and enable Internet connectivity and possibly Internet-based telephony. The set top functionality is enabled through specialized hardware and software.

In the past, Set Top software provided relatively simple functionality, was small and unstructured, and concerned primarily with minimizing the required memory capacity and processing cycles. However, the support of services such as data, Internet, Internet Telephony, etc. poses a new set of problems, including secure access by multiple users, multiple application management, management of resources, decoding, composition, coordination and presentation, of audio, video, graphics, and other data, etc. Existing Set Top software solutions do not solve these problems.

Accordingly, it would be desirable to provide Set Top software that can progress in an evolutionary process from a relatively simple functionality, small, unstructured, memory- and processing cycle-saving software, to software that has a relatively complex functionality, and is larger, structured, and memory- and processing cycle-intensive. Preferably, the fundamental technology underlying the next generation Set Top software architecture employs Java(tm), ActiveX(tm) or an equivalent type of component based object-oriented technology.

The technology should allow the consideration of Set Top application and middleware software functionality independent from the operating system and hardware.

Moreover, it would be desirable to provide an apparatus and machinery for support of multiple users, secure controlled access of resources, application download, registration, start, stop, and monitoring, resource download, registration, start, stop, and monitoring, management of audio, video, data presentations, and some additional functionality.

The present invention provides a system having the above and other advantages. We refer to this technology as the General Instrument "Set Top Applications and Middleware Platform (STAMP)".

SUMMARY OF THE INVENTION

The present invention provides a software architecture framework for the soft set top.

The Set Top software allows the consideration of Set Top application and middleware software functionality independent from the operating system and hardware.

The invention can support various type of functionality, such as multiple users, secure controlled access of resources, application download, registration, start, stop, and monitoring, resource download, registration, start, stop, and monitoring, and management of audio, video, and data presentations.

In a particular implementation, the invention is used to provide a television set-top terminal with software. The terminal includes a computer readable medium having computer program code means, and means for executing the computer program code means to implement a layered software architecture. In the layered software architecture, an application layer allows a user to interact with the terminal, a middleware layer supports the application layer by providing Application Program Interfaces (APIs), an operating system layer supports the middleware layer, and a hardware layer supports the operating system layer. Additionally, the layered software architecture allows configuration of a functionality of the application layer and the middleware layer independently of the operating system layer and the

hardware layer.

The layered software architecture includes a set top management layer that supports the application layer by configuring management services of the terminal. The management services include at least one of application, user, resource and presentation management.

The set top management layer may implement a state information module to designate states of resources of the terminal. The state information module may be based on the ITU-T X.731 standard.

The software architecture may further provide an application program interface (API) for providing a configurable functionality. The API may enable the terminal to support multiple users, or to secure controlled access of resources. The API may enable the terminal to download, register, start, stop, and monitor applications of the applications layer. The API may also enable the terminal to manage audio, video and/or other data presentations.

The software architecture may also provide one or more of: a set top manager, a presentation manager, an application manager, a user manager, a resource manager, a set top agent, and a program view assistant.

A corresponding method is also presented.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a set top terminal software architecture evolution in accordance with the present invention.

FIG. 2 illustrates a detailed set top terminal software architecture in accordance with the present invention.

FIG. 3 illustrates a set top management top-level class diagram in accordance with the present invention.

FIG. 4 illustrates a set top application manager top-level class diagram in accordance with the present invention.

FIG. 5 illustrates a set top applications management relations top-level class diagram in accordance with the present invention.

FIG. 6 illustrates a set top control application top-level class diagram in accordance with the present invention.

FIG. 7 illustrates a set top user manager top-level class diagram in accordance with the present invention.

FIG. 8 illustrates a set top resource manager top-level class diagram in accordance with the present invention.

FIG. 9 illustrates a top-level class diagram of available set top devices in accordance with the present invention.

FIG. 10 illustrates a set top software architecture framework in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a software architecture framework for the soft set top.

FIG. 1 illustrates a set top terminal software architecture evolution in accordance with the present invention.

Set Top software is seen as an evolutionary process that progresses from relatively simple functionality, small, unstructured, memory- and processing cycle-saving to relatively complex functionality, larger, structured, memory and processing cycles intensive. Java(tm), ActiveX(tm) or an equivalent type of component based object-oriented technology is seen as the fundamental technology underlying next generation Set Top software architectures.

We refer to this technology as the "General Instrument Set Top Applications and Middleware Platform (STAMP)". This technology allows the consideration of Set Top application and middleware software functionality independent from the operating system and hardware.

STAMP provides an apparatus and machinery for support of multiple users, secure controlled access of resources, application download, registration, start, stop, and monitoring, resource download, registration, start, stop, and monitoring, management of audio, video, data presentations, and some additional functionality. This technology allows the consideration of Set Top application and

middleware software functionality independent from operating system and hardware.

STAMP is architected based on the Java API platform. Therefore, its most straightforward STAMP realization is using Java. FIG. 1 illustrates the Set Top software architecture evolution in a simplified form. A present architecture includes a hardware layer 100, an operating system layer 110, and a native applications and middleware layer 120. The "native" applications and middleware are designed for use with a specific operating system and hardware environment. In a near-term future architecture, the applications and middleware layer 120 evolves to include a combination of proprietary applications and middleware 122, and the native applications and middleware 120. In a long-term future architecture, the applications and middleware layer 120 evolves to include only proprietary applications and middleware 122, which are independent of the operating system and hardware environment. For example, an application such as a browser is written to an abstraction layer that is independent of the operating system and hardware environment.

The approach taken in the design of the STAMP is to use STAMP components unless it is not feasible due to performance or other requirements, in which case the STAMP components can be replaced with native components.

FIG. 2 illustrates a detailed set top terminal software architecture in accordance with the present

invention. The Figure illustrates the layered architecture of the STAMP near-term solution.

The architecture 200 includes a hardware layer 205, a device driver layer 210, and a real-time operating system layer 215. Another layer 222 includes STAMP interpreter tasks 220, compiled STAMP tasks 225, and native application tasks 230. A next layer 232 includes native APIs 265 and a number of sublayers, including a STAMP components sublayer 260, a STAMP application APIs sublayer 255, a standards API sublayer 250, encompassing standards such as DAVIC, DVB, ATSC, etc., a STAMP middleware APIs sublayer 245, and a sublayer with STAMP interpreter tasks 235 and a STAMP compiler 240.

Next, a STAMP set top management layer 270 is provided, then a layer 272 with STAMP applications 275 and native applications 280. An applet layer 285 is also provided.

The layers within the software architecture indicate possible dependency from higher layer to lower layer. For example, an application will use STAMP components 260, which in turn use some standard APIs 250, which may run on a STAMP interpreter 235 or may be compiled on a STAMP compiler 240.

Generally, each layer is said to support or service the layers above it.

The user interacts with the Set Top through the two topmost layers, the applet layer 285 and the application layers 275, 280. Any type of interaction with the set top is through applications

or applets. The different layers have been defined with the following reasoning:

1. The topmost layer contains processing entities which are temporary in nature and run within applications, e.g., such as a browser, and are downloaded over broadcast channels, over the Internet, or are distributed otherwise. Applets run within applications, and therefore take advantage of the applications' resources and can be restricted in their capabilities. Applications can be STAMP applications or native applications.

2. Applications are permanent, or at least long-term. They are the only unit-level application processing entities and are part of the basic Set Top software distribution or some later upgrade through code download or other means. On the STAMP side, they are implemented as lightweight process groups (e.g., Java thread groups).

3. STAMP Set Top Management controls all Set Top resources and multiplexes them between the different Set Top applications.

4. Applications, applets, and management components are built out of STAMP component sets (e.g., Java Bean Bags). Different component sets are defined for different problem domains, e.g., graphics, networking, etc. The advantage of using component sets is the uniformity of components, i.e., it is easier to build a house out of bricks than variable size stones.

5. STAMP components are built using a variety of APIs. The topmost API layer in the software

architecture consists of those APIs that have been defined to create suitable abstractions for different domains such as conditional access, set top management, etc. STAMP components can use any of the API layers, but the preferred API access layer is the next lower one, which is the STAMP Applications API layer 255.

6. STAMP Applications APIs are defined using Standard APIs which define industry-wide viable abstractions of common Set Top functionalities such as MPEG PSI, DVB SI, etc. STAMP Application APIs can use any of the API layers, but the preferred API access layer is the next lower one, which is the Standard API layer 250.

7. Standard APIs are defined using STAMP Middleware APIs 245, e.g., Standard Java APIs such Personal Java, Java Security API, etc.

8. STAMP Middleware APIs execute on an interpreter (such as a JVM) or are compiled.

9. If STAMP Middleware APIs are interpreted, the interpreter executes as a task or a group of tasks within the Operating System 215.

10. The Operating System 215 controls all the Set Top resources and enables interaction between the various tasks and resources.

11. The Operating System 215 uses Device Drivers 210 to control and communicate with a variety of devices.

12. The bottom layer is the actual hardware.

Agents are processing units that enable functionality on behalf of another entity. Examples

of agents might be a "Management" agent that allows the IRD to be remotely managed by some headend facility either via MPEG transport, modem, or other method. Or, a "Personal" agent might act on a user's behalf to monitor the user's actions and search for material that matches their behaviors (e.g., ProgramViewAssistant).

§1. Execution and Management Framework

STAMP assumes the following sequence of steps after a Set Top is turned on:

1. After the OS is up and running, the STAMP Interpreter is started with the GI SetTop Program.

2. The GI SetTop Program consists of a single infinite loop which initializes and starts the processing threads.

- The first threads started are the "management threads", which service all other threads. These threads also catch exceptions thrown by initialized devices.

3. Application Threads are started as a result of actions by the SetTop Manager Thread Group.

4. Applets are started by Applications.

5. The SetTop Manager Thread Group control all tasks on the Set Top, inside or outside of the interpreter.

6. The SetTop Manager Thread Group can stop the JVM and the Set Top itself.

The Set Top Management Framework consists of the following management entities:

1. The Set Top Manager (SetTopManager) - controls the Set Top and initiates and starts all functions; configures all management services including application, user, resource, and presentation management; closely interacts with the presentation manager, which controls the presentation of all audio-visual materials; configures, starts, stops, and interacts with the Set Top Agent which enables remote Set Top management through the Internet or over the transport stream; configures, starts, stops, and interacts with the Program View Assistant.

2. Presentation Manager - consists of three types of elements, namely decoders, compositors, and presenters; configures all presentable devices (video, audio, etc.); controls the decoding of all presentable materials; composes all presentable materials for audio and/or video presentation; presents all presentable materials; possibly interacts with event processing and forwarding to all registered applications; supports a user interface for presentation preferences.

3. Application Manager - manages the life-cycle of applications including, registration (loading) of applications, maintenance of a list of applications available, possibly decryption or decompression and move to address space, assessment of whether application can run based upon its execution attributes, start/stop/monitor application treads and allotted resources, vehicle for intra-application communication, propagates events to

applications that care (registered); and supports a user interface for application management.

4. User Manager - manages a list of users, their preferences, and their access rights with regards to set top resources. Verifies authorizations / security to enable components to operate. Supports a user interface for user preferences, user management, and access control management of the set top.

5. Resource Manager - Provides access to all set top resources through a variety of resource specific APIs. Manages those resources. Supports a user interface for resource management.

6. Set Top Agent - A Java SNMP Agent with a MIB that permits fault, configuration, accounting, performance, and security management of the Set Top.

7. Program View Assistant - An intelligent agent utilizing forward chaining based on the unification algorithm; minimally, it allows the association of users with different types of presentation materials using the IPG database; it supports an interactive display that can be started as an application or assists the user in background mode through alerts, operations short cuts, and other types of help.

The figure discussed below is a Unified Modeling Language (UML) class diagram depicting the class relations between various set top management components.

FIG. 3 illustrates a set top management top-level class diagram in accordance with the present

invention. FIG. 3 uses the Unified Modeling Language, developed by Rational Software Corporation, USA. A class diagram represents the static structure of a system, and shows a pattern of behaviors that the system exhibits. This is accomplished by showing the existence of classes and their relationships. Each class is represented by a box with three sections. The top section lists the class name. The middle section denotes a list of attributes, and the bottom section denotes a list of operations. In the figures, only the class names are shown for simplicity.

A solid line between classes denotes an association. A solid line with a white diamond tip denotes aggregation by reference, while a black diamond tip denotes aggregation by value. A triangular arrowhead denotes a restricted navigation, e.g., inheritance of operation but not of structure.

Additionally, a number may optionally be shown next to a line end that touches a class box to indicate the cardinality of the relationship, e.g., 0, 1 . . . Also, "*" denotes "or more", e.g., "0. . . .*" denotes "0 or more". "0. . .1" denotes an aggregation of 0 or 1.

All management entities consist of processing threads (lightweight processes or tasks). All of the management processing threads are part of the SetTopManager ThreadGroup 305, which is further subdivided into the ApplicationManager 310, PresentationManager 315, UserManager 320,

ResourceManager 325, and SetTopAgent 330 ThreadGroup(s).

All management entities interface the set top management information base (SetTopMIB) 335. Each management entity (MngmtEntity) 355 controls a part of the MIB while the SetTopAgent 330 provides external access to the MIB (i.e., SNMP, RMI, CORBA, IIOP, etc.). MngmtEntity 355 is associated with a Thread 360 and a ThreadGroup 365, both from java.lang.

The ApplicationManager 310 controls the applications registry (AppRegistry) 340, the presentation manager controls the PresentationRegistry 345, the user manager controls the user registry (UserRegistry) 350, and the resource manager controls the resource registry (ResourceRegistry) 355. The following subsections outline the different management entities in more detail.

Note that the invention enables the application of generic state information modules, such as those described in the ITU-T X.731 standard, within applications, resources, and the set-top MIB. This standard is part of the Q3 standard adopted by the Telecommunications Management Network (TMN). The TMN standard was developed by the CCITT (now the ITU-T) to provide an architecture to achieve an interconnection between various types of management systems and telecommunications equipment for the exchange of management information over standardized

interfaces. TMN is largely based on OSI management standards and includes:

Principles for TMN (ITU-T M.3010 and M.3020), which defines the architecture;

Generic Network Information model (ITU-T M.3100);

Management Services (ITU-T M.3200);

Management Function (ITU-T M.3400); and

Protocol Profiles for Management Interfaces (ITU-T Q.811, Q.812, Q.773).

It is intended that all networks, telecommunications services, and major type of equipment may be managed by TMN.

TMN functions exchange management information by means of the ITU-T X-700 series (OSI system management) standards. Each software component in a TMN layer represents itself and the resources it manages to the layer above as a managed object. The interactions between manager and agent are defined by means of CMISE/CMIP. The organization of the information architecture, the MIB, contains managed objects for specific technologies that can be refined from the general template provided in ITU-T M-3100, the Generic Network Information Model.

All TMN communication is based on the Agent-Manager paradigm. The Q3 interface relies on the OSI management model using the OSI Common Management Information protocol. CMIP is used between Common Management Information Service Elements (CMISE) to provide Common Management Information Service (CMIS).

§1.1. Applications Management

The application manager controls a number of applications which are all built based on the generic applications template which enables the monitoring and control by the application manager.

The *application management* APIs provide the system with application life-cycle and application integrity related functions.

Application Verification and Validation API - The application verification and validation API provides the system with functions for TBD.

Application Life Cycle API - The application life cycle API provides the system with functions for loading, starting, pausing, stopping, and unloading applications.

Application Registration API - The application registration API provides the system with functions for registering application version and application source information.

FIG. 4 illustrates a set top application manager top-level UML class diagram in accordance with the present invention.

Like-numbered elements correspond to one another in the various figures.

Here, a ResourceClient class 405 is associated with a GenericApp class 410, which is associated with a Modem Proxy class 415, which is associated with a ResourceProxy class 420.

The ResourceClient class 405 and ResourceProxy class 420 may be from org.davic.resources. The

GenericApp class 410 may be from gi.settop.applications.

Generic applications implement resource client 405 and resource proxy 420 interfaces for all resources used by the individual applications 410.

All applications, although managed by the application manager only, have to interact with the presentation 315, the user 320, and the resource manager 325. FIG. 5 contains a UML class diagram illustrating these dependencies.

FIG. 5 illustrates a set top applications management relations top-level UML class diagram in accordance with the present invention.

The GenericApp class 410 may be associated with various classes, such as a Download class 505, a WatchTV class 510, an Email class 515, a Ticker class 520, a WebBrowser class 525, and an EPG class 530, each of which may be from gi.settop.applications.

The interaction between the user and all management entities is accomplished through the set top control application, which, as are all other applications, is controlled by the application manager 310 through the generic application 410 template.

FIG. 6 illustrates a set top control application top-level UML class diagram in accordance with the present invention.

The set top control application (SetTopControlApp) 605 coordinates the interaction between the user through a set top control GUI

(SetTopControlGUI) 610, and the different set top managers 310, 315, 320 and 325.

The SetTopControlApp 605 and SetTopControlGUI 610 may be from gi.settop.applications.

§1.2. User Management

The *user profile* APIs provide the system with user-centric and account-centric functions.

Billing API - The billing API provides the system with functions for managing accounts and performing account-related transactions.

Preferences API - The preferences API provides the system with functions for managing user-centric properties.

Favorite Services API - The favorite services API provides the system with functions for managing user-defined sets and subsets of services.

The *security* APIs provide the system with functions for controlling and managing access to applications, content, and services.

Content Control API - The content control API provides the system with functions for controlling access to applications, content, and services based on user-centric criteria (parental locks).

Conditional Access API - The conditional access API provides the system with functions for controlling access to services based on service restrictions and account restrictions.

Authentication and Certificate Management API - The authentication and certificate management API

provides the system with functions for TBD.

Cryptography API - The cryptography API provides the system with functions for encrypting and decrypting content broadcast to the system, stored on the system, or sent from the system.

User access control of the set top and user preferences management is handled by the user manager. The user manager uses the set top MIB as its information store and interacts with the users through the set top control application.

FIG. 7 illustrates a set top user manager UML top-level class diagram in accordance with the present invention.

A SecurityManager class 705 may be from java.lang.

§1.3. Resource Management

The resource management APIs provide the system with functions for measuring, allocating, and de-allocating the user of constrained hardware systems and services.

System Profile API - The system profile API provides the system with functions for determining system resource and configuration information.

Synchronization API - The synchronization API provides the system with functions for synchronizing disparate resources such as audio and video decoders.

Resource Allocation API - The resource allocation API provides the system with functions

for requesting, accessing, and releasing hardware systems and services.

System Test API - The system test API provides the system with functions for external applications to test hardware systems and services.

System Diagnostic API - The system diagnostic API provides the system with functions for built-in-test and diagnosing system faults.

FIG. 8 illustrates a set top resource manager UML top-level class diagram in accordance with the present invention.

A GenericDevice class 805, which may be from gi.settop.devices, is associated with the ResourceManager class 325.

The access to all set top resources by the set top applications is managed through the resource manager 325. The resource manager may use the DAVIC Resource API.

FIG. 9 illustrates a UML top-level class diagram of available set top devices in accordance with the present invention.

The GenericDevice class 805 may be associated with one or more of the following example classes: HardDrive 902, DiskDrive 904, RAMDisk 906, SmartCard 908, DVD 910, DC2Sat 912, ATSCTerminal 914, MPEG 916, MPEG Video 918, MPEG/AC3Audio 920, Terminal 922, DVBSat 924, Keyboard 926, InputDevice 928, HSSIO 930, VideoOut 932, AudioOut 934, FrontPanelDisplay 936, SODA 938, IEEE1284 Parallel, UHFRemote 942, InfraredRemote 944, IEEE1394 946,

AnalogAudio 948, S/PDIG Digital Audio 950, Communications 952, DigitalCable 954, XDSL 956, V34 958, 10BaseT 960, ATM 962, SMPTEVideoOut 964, RGBVideoOut 966, and CCIR16VideoOut 968.

Each of the classes may be from gi.settop.devices.

Available resources are being interacted with through the concept of a generic device. The *device control* APIs provide the system with functions for managing devices. Media control APIs include:

Tuning and Demultiplexing API - The tuning and demultiplexing API provides the system with functions for selecting services and streams.

Decoder API - The decoder API provides the system with functions for decoding video streams, audio streams, data streams, and elements within those streams.

Playback Control API - The playback control API provides the system with functions for managing the playback of streams.

§1.4. Presentation Management

Presentation management and the Presentation API play a role in two different types of scenarios within the Set Top:

1. Presentable objects (graphics, video, audio) being created as a result of an application activation or application related processing.

2. Presentation objects being created as a result of a decoding process of data coming over a carrousel, over the Internet, etc.

The Presentation API and Presentation Manager (PM) facilitate both scenarios in the same manner, i.e., the same object is created and has the same methods that can be activated.

The *presentation* APIs provide the system with functions for displaying and controlling the display of information, status, content, and control to the user.

Video Presentation API - The video presentation API provides the system with functions for managing the display of video data to the primary display surface.

Front Panel Display Manager API - The front panel display manager API provides the system with functions for managing the display of information, status, and control to front panel display surfaces.

Graphics Presentation API - The graphic presentation API provides the system with functions for managing the display of graphic data to the primary display surface.

Font Management API - The font management API provides the system with functions for loading, selecting, and displaying fonts on the primary display surface.

Audio Presentation API - The audio presentation API provides the system with functions for managing the playback of audio data to the primary audio outputs.

A number of applications and applets within STAMP have GUIs. These include:

1. Installation & Diagnostic screens;

2. Set Top option screens which provide access to major Set Top features and applications and implement configuration screens (e.g., rating preferences, language preferences, audio/video modes, etc.);

3. Banners and detailed information for Watch TV;

4. IPG screens;

5. Web Browser screens;

6. E-mail screens;

7. Ticker (e.g., for stock prices);

8. Applet Screens; and

9. Downloadable application screens.

All these GUIs are built out of a set of components which are defined through the Presentation API.

§2. Applets

Java Applets are supported on Set Tops that host either a Java enabled Java Browser or a Java enabled Native Browser. The applet is downloaded via http over the Internet or via a DSMCC carrousel service.

§3. Applications

§3.1. Watch TV

Watch TV - The system provides a simple broadcast television viewing environment. The user may channel up, channel down, return to a previous channel, and select a specific channel.

Watch TV with Parental Locks - The system may provide a means for the user to limit access to services, channels, or events. A password mechanism may over-ride the access limitations. Access may be based on channel, event rating, event theme, or some other criteria.

Watch TV with Conditional Access - The system may provide a means to limit access to services, channels, or events based on the access rights of the user as a customer of the service provider. The access rights may pertain to service limitations or account limitations.

Watch TV with Alternate Audio- The system may make available more than one audio stream for a specific video stream. The user has the opportunity to select an alternative audio stream for playback in synchronization with the video stream.

Watch TV with Subtitling- The system may make available one or more text or graphic streams that may be synchronized and superimposed upon the video stream.

§3.2. Downloading Applications

The *downloading application* scenarios describe situations in which a broadcast application or data service is being bought or stored on the system.

Watch TV with Persistent Unsynchronized Data Service - The system provides a data service that is not synchronized to the current audio and video stream (for example, a stock ticker). The application that uses the data service (for example,

the stock ticker graphical user interface) is already resident on the system, only the data (the stock quotes) is transmitted in this scenario.

Watch TV with Persistent Synchronized Data Service - The system provides a data service that is synchronized to the current audio and video stream (for example, sport statistics that correspond to the player currently on screen). The application that uses the data service is already resident on the system, only the data is transmitted in this scenario.

Watch TV with Scheduled Download of Application - The system provides a means of scheduling and subsequently downloading an application located on a service that may not be the current service.

Watch TV with Impulse Buy of Application - The system provides a means for the user to be notified of an available application and a means for the user to purchase and download the application.

Watch TV with Background Application Update - The system provides a means for the user to download an application in the background (without disrupting the current audio and video presentation).

§3.3. Teleshopping

The *teleshopping* scenarios describe situations in which the user may request information, products, or services through applications made available on the system.

Watch TV and Request Information - The system provides a means for the user to request information

that corresponds to the currently broadcast video and audio. For example, during an automobile commercial, the user may request a brochure.

Watch TV and Impulse Buy - The system provides a means for the user to purchase a product that corresponds to the currently broadcast video and audio. For example, during a commercial for a commemorative plate, the user may purchase the plate.

Browse and Buy from a Video-centric Catalog - The system provides a means for the user to navigate through a catalog that contains graphics, video, and audio. The user can select and purchase a product or service in the catalog.

Browse and Buy from a Audio-centric Catalog - The system provides a means for the user to navigate through a catalog that contains graphics and audio. The user can select and purchase a product or service in the catalog.

Browse and Buy from a Graphics-centric Catalog - The system provides a means for the user to navigate through a catalog that contains graphics. The user can select and purchase a product or service in the catalog.

§3.4. Near Video on Demand

The *near video on demand* (NVOD) scenarios describe situations in which the user can select movies (or some other audio-video service) from a multiplex of similar movies or service.

Pay-per-View - The system provides a means for the user to purchase a service from a multiplex of services. Interactivity is generally limited to the selection and purchasing process.

Interactive Pay-per-View - The system provides a means for the user to purchase a service from a multiplex of services. The user can "pause" the service until the next availability of the service.

§3.5. Gaming

The gaming scenarios allow a user to play a game via a network-delivered service. Games may be single-user or multi-user games either located entirely on the system (local) or played over a network.

Watch TV and Local Participate - The system provides a means for the user to watch a television program and "play along" with a single-user local application. For example, during "Wheel of Fortune" the user can "spin the wheel" and make their own guess.

Watch TV and Multi-user Participate - The system provides a means for the user to watch a television program and "play along", competing with other users.

Single Player Game - The system provides a means for the user to play a system-based game with no additional remotely located participants.

Multi-user Game - The system provides a means for the user to play a system-based game in which there is additional, remotely located, participants.

§3.6. Internet Access

The *internet access* scenarios allow the user to access internet-based content or functions: html pages, email, chat, streaming audio, etc.

Browse the Web - The system provides a means for the user to browse web-sites.

Email - The system provides a means for the user to read, write, and respond to electronic mail.

Chat - The system provides a means for the user to communicate in real-time with other users on the Internet.

Watch TV with Internet Content - The system provides a means for the user to simultaneously watch an audio and video service while performing an Internet-based activity that is made available with or through the audio-video service.

Browse Web with TV Content - The system provides a means for the user to watch an audio or video service that is made available from an Internet-based item.

§3.7. Information on Demand

The *information on demand* scenarios allow the user to access information-related content when they desire or customized to their preferences.

Pay-per-View News Service - The system provides a means for the user to purchase an audio, video, or text-centric news service. The service might provide a headline-level of detail, an in-depth level of detail, editorial information, or some other level of service.

Query for News Item - The system provides a means for the user to query multiple available sources for one or more "articles" pertaining to user-defined or user-selected criteria.

Personalized News - The system provides a means for the user to tailor a news service to their preferred topics.

Subscription Services - The system provides a means for the user to purchase and schedule a subscription to an audio, video, or text-centric news service.

§3.8. Electronic Program Guide

The *electronic program guide* scenarios allow the user to access channel, service, and event information. Generally the user may also control the system or navigate within the system using this information.

Current Channel Now and Next - The system provides information on what service is currently available on the current channel and what service is following.

All Channels Now and Next - The system provides information on what service is currently available on all known channels and what services are following.

All Channels Now and Future - The system provides information on what service is currently available on all known channels and what services are available for some predetermined time in the future.

Request Preview from Guide - The system provides a means for the user to request a sample of a service. For example, the user might be able to request a trailer for a motion picture or a demo of an application.

Purchase Event from Guide - The system provides a means for the user to purchase a future event or service from within the electronic program guide.

Setup Program Reminder - The system provides a means for the user to schedule a notification of a future event at a future point in time.

Purchase Application from Guide - The system provides a means for the user to purchase an application for download from within the electronic program guide.

§3.9. Distance Learning

The *distance learning* scenarios describe a virtual classroom in which students and teachers are not co-located.

Watch TV with Chat - The system provides audio and video of a virtual classroom. The system provides a means for the user to participate with an educator or other students (located at different sites) through chat.

Watch TV with Test - The system provides audio and video of a virtual classroom. The system provides a means for the user to participate with an educator or other students (located at different sites) through chat.

§3.10. Home Banking

The *home banking* scenarios describe services available to the user comparable to services available at a retail bank.

Information Retrieval - The system provides a means for the user to access information pertaining to banking functions and their personal accounts.

Transaction - The system provides a means for the user to perform typical retail bank electronic transfers.

Service Applications - The system provides a means for the user to apply for retail bank services such as credit cards, loans, and checking.

Counseling - The system provides a means for the user to get personalized financial assistance.

§3.11. Navigation

The *navigation* scenarios describe situations in which the user activates services, applications, and content available on the system.

Application Activation - The system provides a means for the user to activate (run or launch) an application.

Application Removal - The system provides a means for the user to remove an application from system memory, local storage, or remote storage associated with the system.

System Preferences - The system provides a means for user to identify user-centric preferences

related to operation of the system or its user interface.

Favorite Channel Setup - The system provides a means for the user to identify or customize the set of or subset of available services.

§4. STAMP Component Architecture

Component Groups (Bean Bags) are defined to facilitate domain specific (e.g., IPG) application component building sets. These set are NOT disjoint. The Java Bean Bags are:

1. GI Application System Bean Bag - this set of components comprises all beans that are generic to all different application component sets. This may include information exchange mechanisms, synchronization mechanisms, timing mechanisms, storage mechanisms, etc.

2. GI GUI Bean Bag - this set of components includes all GUI beans that are generic to all different applications; this may include lists, buttons, sliders, etc.

3. GI HotJava Bean Bag - this set of components includes all components that can be a part of a browser.

4. GI IPG Bean Bag - this set of components includes all components that can be a part of the IPG application.

5. GI Tuning Bean Bag - this set of components includes all components that can be a part of a tuning application.

6. GI E-mail Bean Bag - this set of components includes all components that can be a part of an e-mail application.

7. GI Download Bean Bag - this set of components includes all components that can be a part of a download application.

8. GI Ticker Bean Bag - this set of components includes all components that can be a part of an information ticker application.

9. GI Game Player Bean Bag - this set of components includes all components that can be a part of a game player application.

§5. Applications Programming Interfaces (APIs)

APIs provide the system with a hardware and operating system independent abstraction layer. This permits broadcast applications to be delivered to systems of widely varying processors and operating systems: from set-top boxes with a RISC processor and a real-time operating system to a personal computer with a microprocessor and a non-real-time operating system.

All APIs are usable from native applications as well as from Java applications. The Following APIs need to be defined outside the scope of those APIs defined by DAVIC and those standardized by JavaSoft:

§5.1. Event Management APIs

The *event management* APIs provide the system with functions for sending and receiving messages, events, and "interrupts."

Event Dispatch API - The event dispatch API provides the system with functions for sending messages, events, and "interrupts."

User Input API - The user input API provides the system with functions for receiving and dispatching user-originated events (keyboard, mouse, front panel, remote control, etc.).

Inter-Process Communication API - The inter-process communication API provides the system with functions for sending messages and events between applications, processes, and threads.

Scheduling API - The scheduling API provides the system with functions for establishing time and date triggered events and "interrupts".

§5.2. Application Utility APIs

The *application utility* APIs provide the system with general purpose functions commonly used by applications.

Math API - The math API provides the system with functions for performing math routines.

Time API - The time API provides the system with functions for accessing time and date and performing common time routines.

String API - The string API provides the system with functions for loading, unloading, and manipulating strings.

List API - The list API provides the system with functions for managing lists and other common dynamic data structures.

File API - The file API provides the system with functions for performing file-related operations.

Localization API - The localization API provides the system with functions for performing common localization operations.

Communications API - High-level interface to all communications and communications management components including dial-up networking, Internet, routing, transport stream, etc.

Object Services API - Support for information exchange between objects; this includes an information model, types, relationships, operations, characteristics, naming, registry, distribution, etc.

Database Services API - support for data store and retrieval mechanisms including a generic file system and registry.

Download Services API - support for download services.

§5.3. Content Management APIs

The *content management* APIs provide the system with content life-cycle and content integrity related functions.

Content Integrity API - The content integrity API provides the system with functions for TBD.

Content Download API - The content download API provides the system with functions for storing applications broadcast to the system.

Content Version Control API - The content version control API provides the system with functions for managing the stored or loaded version of content.

Content Storage API - The content storage API provides the system with functions for storing and retrieving content in the system or with remote storage.

§5.4. System Information APIs

The *system information* APIs provide the system with MPEG-2 system information and program information stream related functions.

Event Information API - The event information API provides the system with functions for interpreting the event information stream.

Channel and Service Map API - The channel and service map API provides the system with functions for interpreting the system and service information streams

§5.5. GI Presentation APIs

See Set Top Management Section. Enables the building of presentation manager components such as the decoders, compositors, and presenters; access to all presentation subsystems including graphics, audio, video as needed in addition to DAVIC and Standard Java APIs; manipulation of all presentation subsystems as needed in addition to DAVIC and Standard Java APIs; initialization, startup, manipulation, shut down and coordination of all presentation components; etc.

§5.6. GI Set Top Management APIs

See Set Top Management Section. This API enables set top management applications by providing component hook-up functionality, component monitoring and control functionality, initialization, startup, shut down and coordination of all internal processes/daemons/applications; processing of asynchronous events as well as events resulting from collecting information; etc.

Operations Support API - facilitates the Set Top Management API through Java/non-Java interfaces as well as an SNMP agent API.

§5.7. DAVIC APIs

The following DAVIC APIs are supported:

1. Resource Notification API (org.davic.resources) - a standard mechanism for applications to register interest in scarce resources and to be notified of changes in those resources or removal of those resources in the environment.

2. The MPEG API (org.davic.mpeg) - includes the MPEG Component API, the MPEG Section Filter API, and the MPEG TransportStream API.

3. The Tuning API (org.davic.net.tuning) - tuning of network interfaces including multiple network interfaces (local and/or remote).

4. The Conditional Access API (org.davic.net.ca) - a CA system independent interface for accessing CA functionality.

5. The DVB SI API (org.davic.net.dvb.si) - an interface to DVB SI database (including the access to the actual transport stream through an optional cache).

6. The ATSC PSIP API (org.davic.atsc.psip) - an interface to the ATSC SI and PSI database (including the access to the actual transport stream through an optional cache).

7. The Media Player API (org.davic.media) - an extension of the Java Media Framework to facilitate Set Tops.

8. The DSM-CC U-U API (davic.dsmccuu) - implements a subset of the DSM-CC-U-U APIs defined in ISO/IEC 13818-6.

§5.8. Standard Java APIs

1. Personal Java - This API is a subset of the JDK 1.1 API, supplemented by a small number of new APIs designed to meet the needs of networked embedded applications. Java APIs introduced after JDK 1.1 will not automatically become a part of the PersonalJava API. New APIs will be reviewed and evaluated for appropriateness before being added to PersonalJava.

The JDK 1.1 packages included in PersonalJava are: java.applet
java.awt
java.awt.datatransfer
java.awt.event
java.awt.image
java.awt.peer
java.beans

java.io
java.lang
java.lang.reflect
java.net
java.util

2. The Java Text API - supports internationalization.

3. The Java Security APIs are a framework for developers to easily and securely include security functionality in their applets and applications. This functionality includes cryptography with digital signatures, encryption, and authentication.

4. Java Commerce API will bring secure purchasing and financial management to the Web. JavaWallet is the initial component, which defines and implements a client-side framework for credit card, debit card, and electronic cash transactions.

5. Java Media Framework (JMF) specifies a unified architecture, messaging protocol and programming interface for media players, capture and conferencing. JMF will be published as three APIs. The Java Media Player will be published first; Java Media Capture and Java Media Conference APIs will be published subsequently. Java Media Player is an API for synchronization, control, processing and presentation of compressed streaming and stored timed media, including video and audio.

6. Java RMI (including CORBA) - object-oriented distributed processing API that supports CORBA.

§6. The Set-Top Software Architecture

FIG. 10 illustrates a set top software architecture framework in accordance with the present invention.

The architecture 1000 corresponds to the architecture 200 of FIG. 2, but provides further details and examples for each layer.

The device driver kernel modules in the layer 210' correspond generally to the classes in FIG. 9.

§7. Relationship to OpenCable APIs

The following Sections summarize the current OpenCable API classification.

§7.1. Operating System <-> Network APIs

Network Protocols are used at all levels of the ASB, to communicate with other clients, applications, and servers. Although applications also make use of network protocols, many of these are actually OS-implemented network stacks called through the OS <-> Application APIs. Protocols specifically dealing with application-level client-server issues will be covered in the Client<->Server section.

§7.1.1. Video and Data Network / Transport Protocols

These protocols are used for network packet routing, connection establishment, control, and teardown. Industry open standards are TCP for connection-oriented transport, UDP for connectionless, real-time traffic transport, and IP for packet routing. RSVP is used for network

bandwidth and channel reservation. MPEG-2 is the clear favorite for video transport.

§7.1.2. Session / Presentation Protocols

These protocols are used for session maintenance, and to provide applications with a common procedure for requesting and receiving data from content servers. FTP is used for file transfer, HTTP for WWW access and HTML transport, NNTP for news server access, and SMTP and IMAP for mail. There is no standard for chat, however.

§7.1.3. Directory Services Protocols

Directories are mostly read-only distributed databases that are designed for attribute lookup to locate, authenticate, and update network objects. LDAP, rapidly emerging as the standard directory for TCP/IP networks, is a lightweight version of X.500's more complex Directory Access Protocol. LDAP has been adopted by over 40 major software vendors, and supports access-independent data views, hierarchical storage, multiple storage schemes, rich access control, multiple database types and instances, and network replication.

§7.1.4. Data-Link, Physical, MAC Layer Protocols

These APIs control modulation, encoding, and management of data streams over the cable plant.

§7.1.5. Network Encryption Protocols

These protocols are used to encrypt and decrypt video, data, and transactions schemes.

§7.1.6. Multi-user / Network Broadcast Protocols

Required for broadcasting of data streams efficiently over the network.

§7.2. OS <-> Applications APIs

The operating system provides applications with two sets of crucial services that allow those applications to be implemented on top of the system. Base OS Services, such as scheduling, virtual memory, multi-threading, etc., are essential services that are provided to almost all applications without much effort on the programmer's part. The APIs for the these services are largely invisible. OS Standard Libraries, on the other hand, are predefined code modules that applications are encouraged to use when implementing functionality supported by the OS. For example, a network application is encourage to use the OS's network libraries, almost all apps are encouraged to use the OS's GUI libraries, etc. In essence, the standard libraries are present to maintain consistency, prevent duplication of effort, and speed up development. Sometimes, applications will bypass certain libraries for aesthetic reasons or to optimize code.

§7.2.1. Application Services and Libraries

These APIs control the heart of the set-top box. Native applications (those written specifically for the ASB, requiring maximum performance and hardware interaction) will be implemented based on

OS services and libraries, provided through these APIs.

§7.2.2. Computer Graphics

These APIs control drawing and rendering of 2D and 3D objects. Application programmers use these to simplify graphic functions. They are also used to implement the graphical user interface and windowing tool-kits for applications.

§7.2.3. Remote Execution Environments / Virtual Machines

Allows applications stored on the network to be downloaded, executed, and discarded. Encapsulates runtime environment, security model, and object model. Current competitors are platform-independent Java, and Windows-specific ActiveX. Scripting versions include JavaScript and VBScript.

§7.3. Operating System <-> Hardware APIs

These APIs, commonly referred to as device drivers, allow the operating system to control various pieces of hardware. They are usually implemented in software by the device manufacturer, who plugs them into the operating system via a well defined device-driver architecture API proposed by the operating-system vendor. For well-defined categories of devices, the operating system vendor will sometimes provide a generic device driver for an entire class of devices. This generic driver can provide lowest common denominator functionality when the manufacturer's driver is unavailable, i.e., the Macintosh OS provides a generic Laser Writer driver for all laser printers, but printers that have their

own driver available can take advantage of vendor / model-specific features. Systems that automatically recognize devices and load the appropriate driver are classified as plug-and-play.

§7.3.1. Hardware Abstraction Layer (HAL) APIs

The HAL separates the hardware from the OS, and makes the OS portable across any platform that supports the HAL.

§7.3.2. Registry APIs

Storage format and control for local Flash RAM storage.

§7.4. Operating System <-> Operating System APIs

These APIs are used when one OS is implemented on top of another, as is the case with virtual machines, or two OSs communicate with each other. The latter case is rare, and is usually handled through network protocols, although some component and object models allow distributed objects to communicate across OSs.

§7.4.1. Remote Execution Environments / Virtual Machines (VM) links to OS

Controls implementation details of virtual machines on existing RTOS. Important for VM portability if a proprietary VM is implemented, or if an open VM is implemented with proprietary ties to the OS.

§7.4.2. Distributed Object Model

Allows applications to store and exchange data with other networked applications in a common object format. Competitors are Microsoft's Distributed

Component Object Model (DCOM) and JavaBeans / CORBA (Common Object Request Broker Architecture). Remote Procedure Call mechanisms (ONC, DCE) also fall within this API.

§7.5. Client <-> Server APIs

Clients and servers will often pass application data through standard networking protocols, but remote client management, authentication, software downloading, and updating are handled through specific client-server APIs. These APIs usually tend to be closed and fragmented, due to the large number of potential combinations and issues.

§7.5.1. Network Management

Remote network management of all network elements.

§7.5.2. Software Management and Updates

An API that controls software downloads and module loaders.

§7.5.3. Network / Remote FileSystem

Controls file storage and retrieval methods.

§7.5.4. Digital Certificates

Control authentication for users, vendors, and products.

§7.5.5. Calendar Server Protocols

Determines storage and interaction formats for user calendaring and scheduling functions.

§7.5.6. Synchronization Protocols

Required for clock synchronization and timing.

§7.5.7. Usage Data Gathering APIs

Defines methods and formats for usage data collection, storage, and transmission.

§7.5.8. Billing System APIs

Defines methods and formats for billing data collection, storage, and transmission.

§7.6. Application <-> File APIs

Applications use specific formats to store and retrieve data that they use. With the exception of a few, high-visibility formats (ASCII, HTML), these formats are often closed and proprietary in nature. Besides just static-data, some applications also support the storage, retrieval, and execution of plug-in code modules, and release APIs that allow 3rd party vendors to construct such plug-ins.

§7.6.1. Text Document Storage Formats

Storage format for plain text documents.

§7.6.2. Hyperlinked Document Storage Formats

Storage format for rich, hyperlinked text.

§7.6.3. Image Storage Formats

Compression format for images.

§7.6.4. 3D Graphics Document Storage Formats

Storage formats for 3D data

§7.6.5. Video Clip Storage Formats

Encoding of streaming and non-streaming video clips.

§7.6.6. Audio Document Storage Formats

Encoding of sound and music.

§7.6.7. Animation Document Storage Formats

Animation encoding.

§7.6.8. Web / Video Integration Formats

Defines libraries and methods to integrate web-based content with traditional TV offerings.

Examples include hyperlinked ads, buttonbar and ticker overlay, virtual web channels, etc.

§7.6.9. Personal Information Storage

Storage of user identification information, preferences, and access control information.

§7.7. Client <-> Content APIs

This set of APIs controls serving and delivery of content from various servers to the client. It is really a subset of the client-server APIs, but is explicitly mentioned since some vendors may attempt to achieve lock-in by requiring proprietary servers to distribute content.

§7.7.1. Web Data Serving

Format for serving web content. Beyond standard HTTP, some web servers may use proprietary serving methods (i.e., Active Server Pages).

§7.7.2. Video Data Serving

Format for serving video content.

§7.7.3. Integrated Data Serving

Serving and coordination of integrated web / data content.

§7.8. Content <-> Access APIs

This set of APIs controls serving and delivery of content from various servers to the client. It is really a subset of the client-server APIs, but is explicitly mentioned since some vendors may attempt to achieve lock-in by requiring proprietary servers to distribute content.

§7.8.1. Video Access Control

Access control and encryption of video streams.

§7.8.2. Data Access Control

Authentication for data content transmission.

§7.9. Browser <-> Resident Engine APIs

Not really a separate set of APIs per se, the browser <-> resident engines interaction is really a sub-category of OS <-> Applications, and is explicitly broken out due to its strategic importance and the fact that most browser vendors subsume this API into their product. In a well-designed operating system, the resident HTML, VRML, Java, and caching engines are services that the OS provides to all interested applications. However, most browser vendors bundle these important engines into the browser, thereby concealing and controlling these important APIs, even though they should be explicitly exposed.

§7.9.1. Explicit HTML Engine Control

Controls to access and use the HTML rendering engine once it has been separated from the browser. Alternatively, if a proprietary format is used, the OS provider should document the APIs used, and provide rights to these APIs for free, in perpetuity.

§7.9.2. Explicit Cache Control

Controls to access and use the caching engine once it has been separated from the browser.

Accordingly, it can be seen that the present invention provides a Set Top software that allows the consideration of Set Top application and middleware software functionality independent from

the operating system and hardware.

The invention can support various type of functionality, such as multiple users, secure controlled access of resources, application download, registration, start, stop, and monitoring, resource download, registration, start, stop, and monitoring, management of audio, video, data presentations.

Although the invention has been described in connection with various specific embodiments, those skilled in the art will appreciate that numerous adaptations and modifications may be made thereto without departing from the spirit and scope of the invention as set forth in the claims.

For example, while various syntax elements have been discussed herein, note that they are examples only, and any syntax may be used.

Moreover, the invention is suitable for use with virtually any type of network, including cable or satellite television broadband communication networks, local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), internets, intranets, and the Internet, or combinations thereof.

What is claimed is:

1. A television set-top terminal with software, comprising:

a computer readable medium having computer program code means; and

means for executing said computer program code means to implement a layered software architecture wherein:

an application layer allows a user to interact with the terminal;

a middleware layer supports the application layer by providing Application Program Interfaces (APIs);

an operating system layer supports the middleware layer;

a hardware layer supports the operating system layer; and

said layered software architecture allows configuration of a functionality of the application layer and the middleware layer independently of the operating system layer and the hardware layer.

2. The terminal of claim 1, wherein:

the layered software architecture includes a set top management layer that supports the application layer by configuring management services of the terminal.

3. The terminal of claim 2, wherein:

said management services include at least one of application, user, resource and presentation management.

4. The terminal of claim 2, wherein:
the set top management layer implements a state information module to designate states of resources of the terminal.

5. The terminal of claim 4, wherein:
said state information module is based on the ITU-T X.731 standard.

6. The terminal of claim 1, further comprising:
an application program interface (API) for providing a configurable functionality.

7. The terminal of claim 6, wherein:
said API enables said terminal to support multiple users.

8. The terminal of claim 6, wherein:
said API enables said terminal to secure controlled access of resources.

9. The terminal of claim 6, wherein:
wherein said API enables said terminal to download, register, start, stop, and monitor applications of the applications layer.

10. The terminal of claim 6, wherein:

said API enables said terminal to manage audio, video and/or other data presentations.

11. The terminal of claim 1, further comprising at least one of:

a set top manager;
a presentation manager;
an application manager;
a user manager;
a resource manager;
a set top agent; and
a program view assistant.

12. A method for implementing a layered software architecture for a television set-top terminal, comprising the steps of:

providing a computer readable medium having computer program code means; and

executing said computer program code means to implement a layered software architecture wherein:

an application layer allows a user to interact with the terminal;

a middleware layer supports the application layer by providing Application Program Interfaces (APIs);

an operating system layer supports the middleware layer;

a hardware layer supports the operating system layer; and

said layered software architecture allows configuration of a functionality of the application layer and the middleware layer independently of the operating system layer and the hardware layer.

13. The method of claim 12, wherein:

the layered software architecture includes a set top management layer that supports the application layer by configuring management services of the method.

14. The method of claim 13, wherein:

said management services include at least one of application, user, resource and presentation management.

15. The method of claim 13, wherein:

the set top management layer implements a state information module to designate states of resources of the method.

16. The method of claim 15, wherein:

said state information module is based on the ITU-T X.731 standard.

17. The method of claim 12, wherein:

said layered software architecture provides an application program interface (API) for providing a configurable functionality.

18. The method of claim 17, wherein:

said API enables support of multiple users.

19. The method of claim 17, wherein:

said API enables secure controlled access of resources.

20. The method of claim 17, wherein:

said API enables downloading, registering, starting, stopping, and monitoring of applications of the applications layer.

21. The method of claim 17, wherein:

said API enables managing of audio, video and/or other data presentations.

22. The method of claim 12, where in said layered software architecture provides at least one of:

a set top manager;
a presentation manager;
an application manager;
a user manager;
a resource manager;
a set top agent; and
a program view assistant.

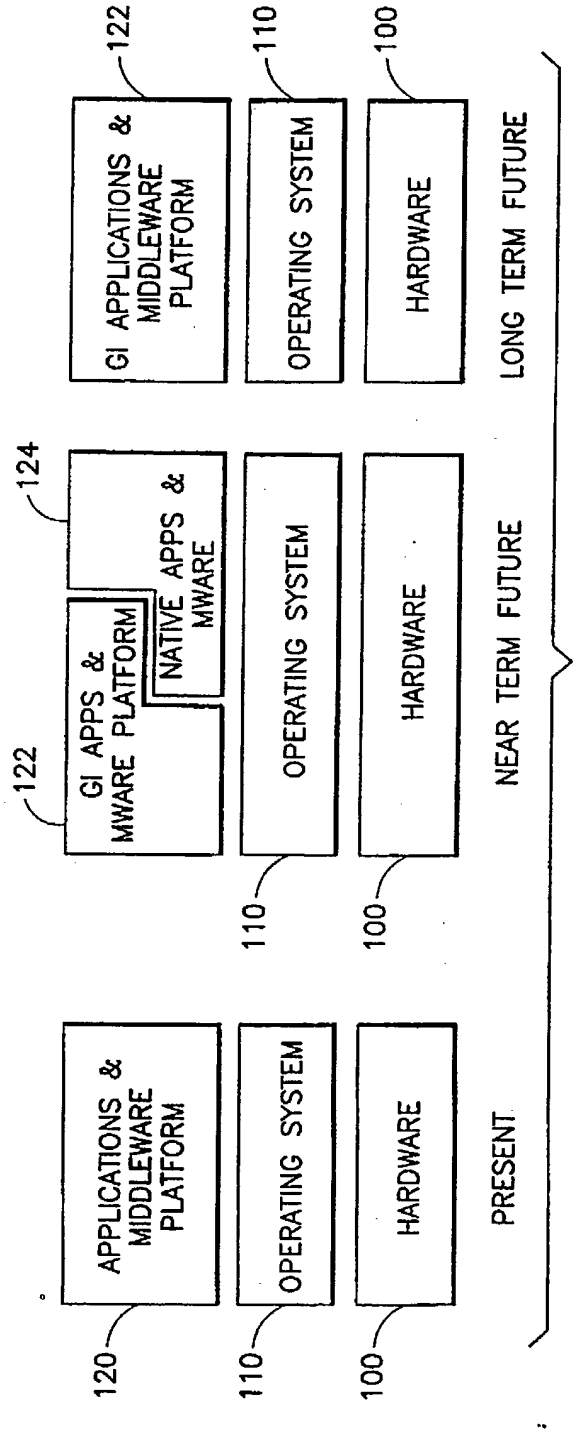

APPLICATIONS & MIDDLEWARE PLATFORM
OPERATING SYSTEM
HARDWARE
120
110
100
PRESENT
GI APPS & MWARE PLATFORM
NATIVE APPS & MWARE
OPERATING SYSTEM
HARDWARE
122
124
110
100
NEAR TERM FUTURE
GI APPLICATIONS & MIDDLEWARE PLATFORM
OPERATING SYSTEM
HARDWARE
122
110
100
LONG TERM FUTURE

FIG.1

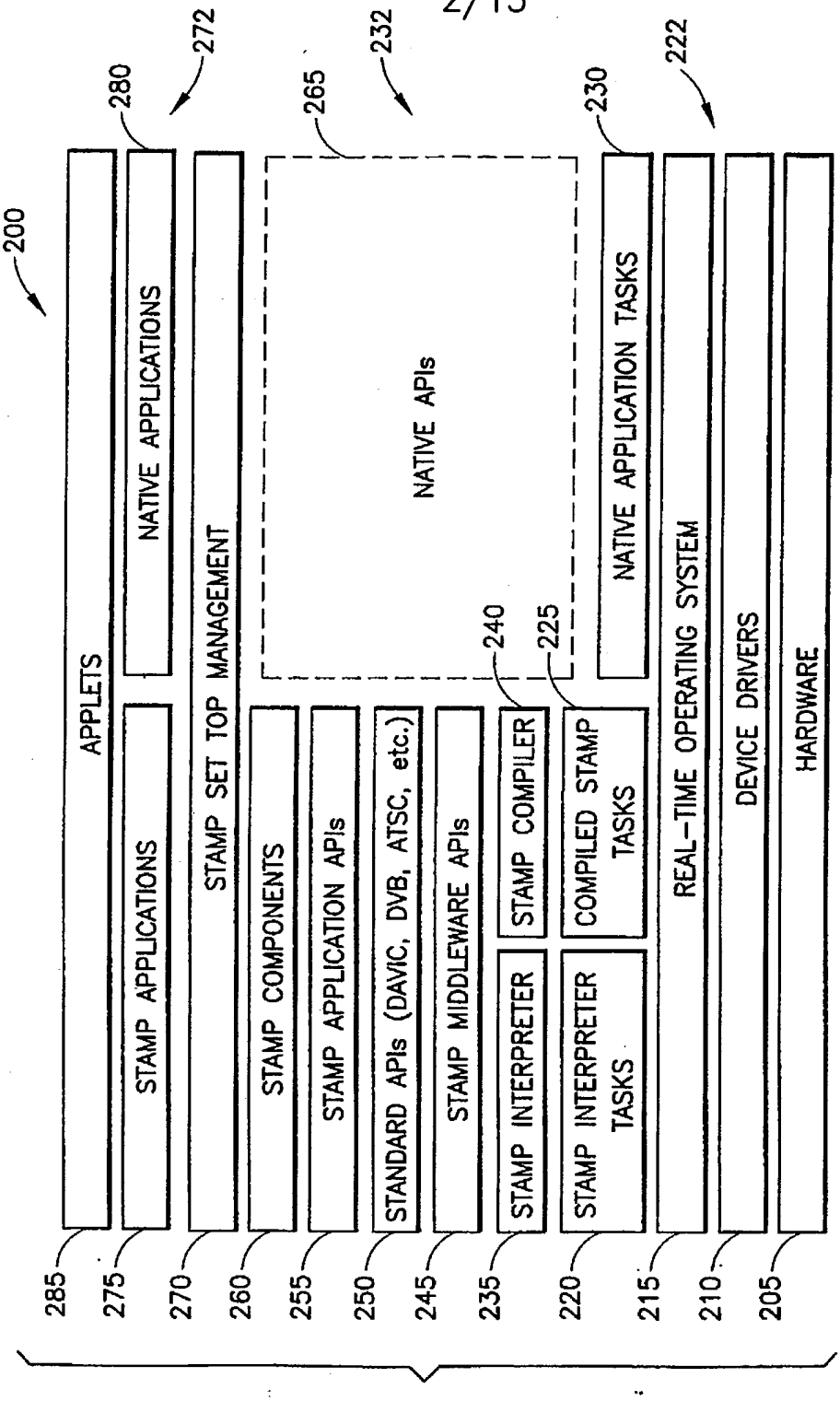
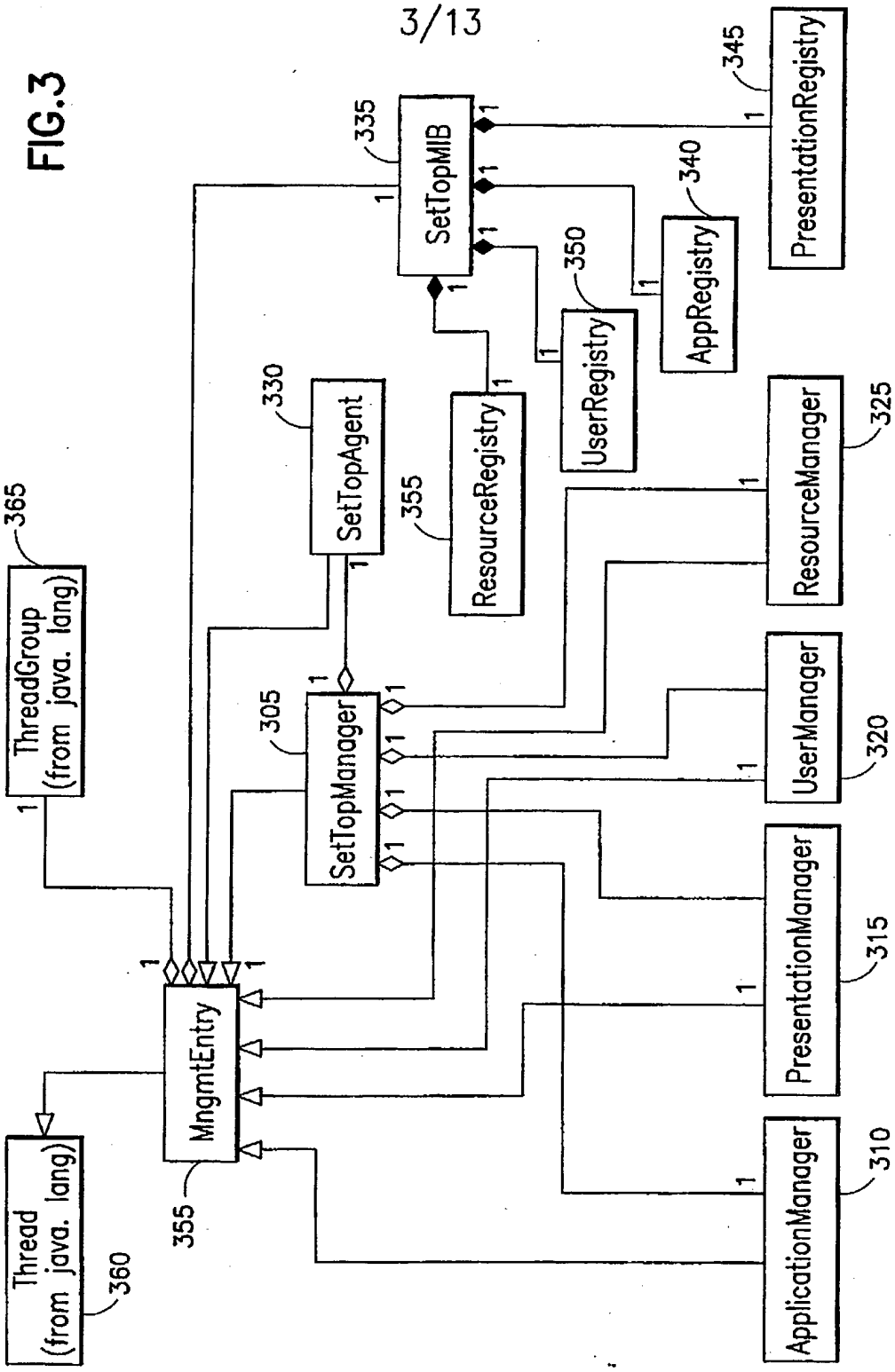

200
APPLETS
STAMP APPLICATIONS
NATIVE APPLICATIONS
STAMP SET TOP MANAGEMENT
STAMP COMPONENTS
STAMP APPLICATION APIs
STANDARD APIs (DAVIC, DVB, ATSC, etc.)
STAMP MIDDLEWARE APIs
STAMP INTERPRETER
STAMP INTERPRETER TASKS
STAMP COMPILER
COMPILED STAMP TASKS
NATIVE APIs
NATIVE APPLICATION TASKS
REAL-TIME OPERATING SYSTEM
DEVICE DRIVERS
HARDWARE
280
272
265
232
230
222
240
225
285
275
270
260
255
250
245
235
220
215
210
205

FIG.2

FIG.3


ThreadGroup (from java. lang)
365
Thread (from java. lang)
360
MngmtEntry
355
SetTopManager
305
SetTopAgent
330
SetTopMIB
335
ResourceRegistry
355
UserRegistry
350
AppRegistry
340
PresentationRegistry
345
ResourceManager
325
UserManager
320
PresentationManager
315
ApplicationManager
310

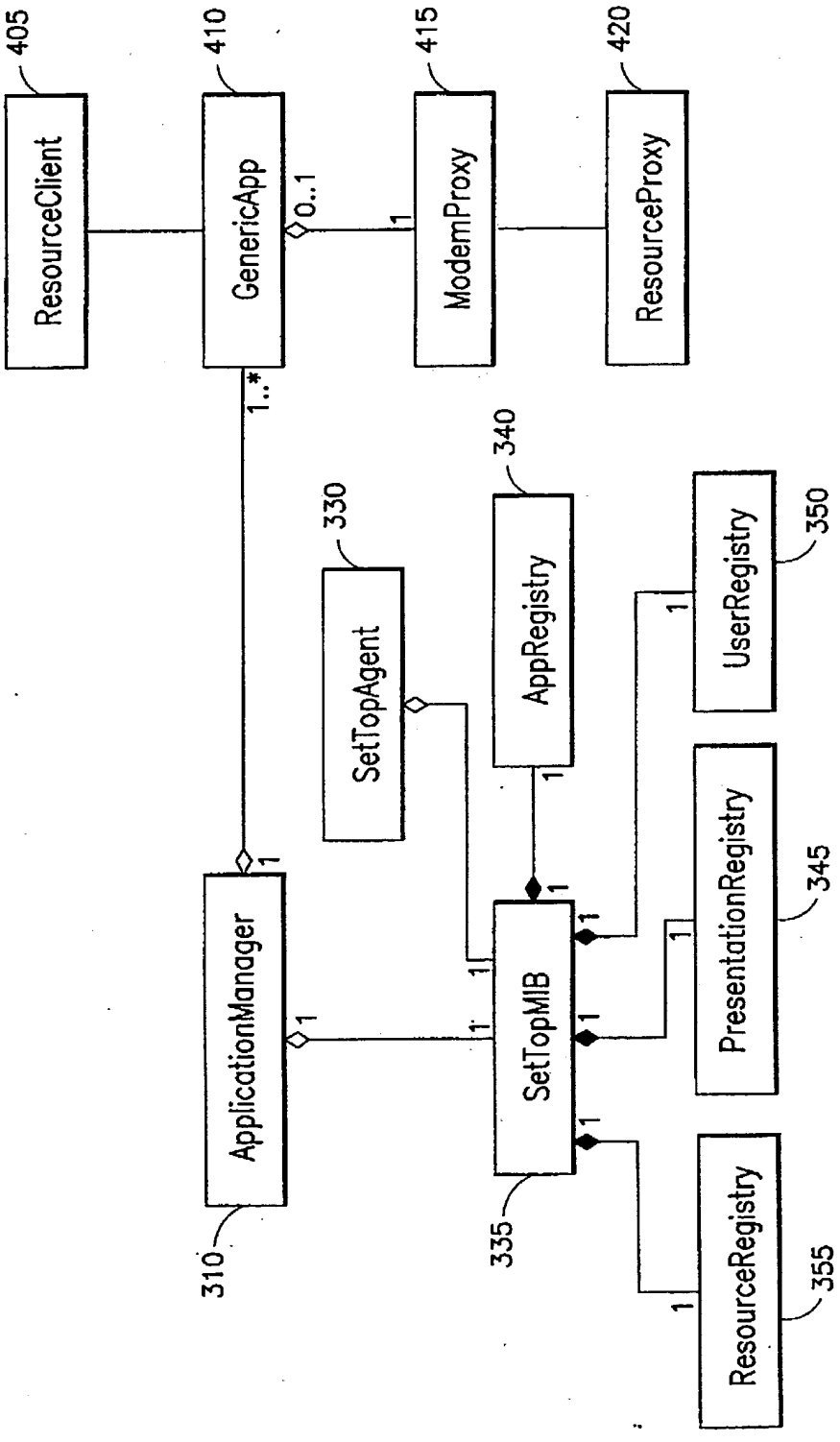

405
ResourceClient
410
GenericApp
0..1
415
ModemProxy
420
ResourceProxy
1..*
1
330
SetTopAgent
340
AppRegistry
350
UserRegistry
ApplicationManager
1
1
1
1
1
1
1
1
1
1
1
1
SetTopMIB
PresentationRegistry
345
310
335
ResourceRegistry
355

FIG.4

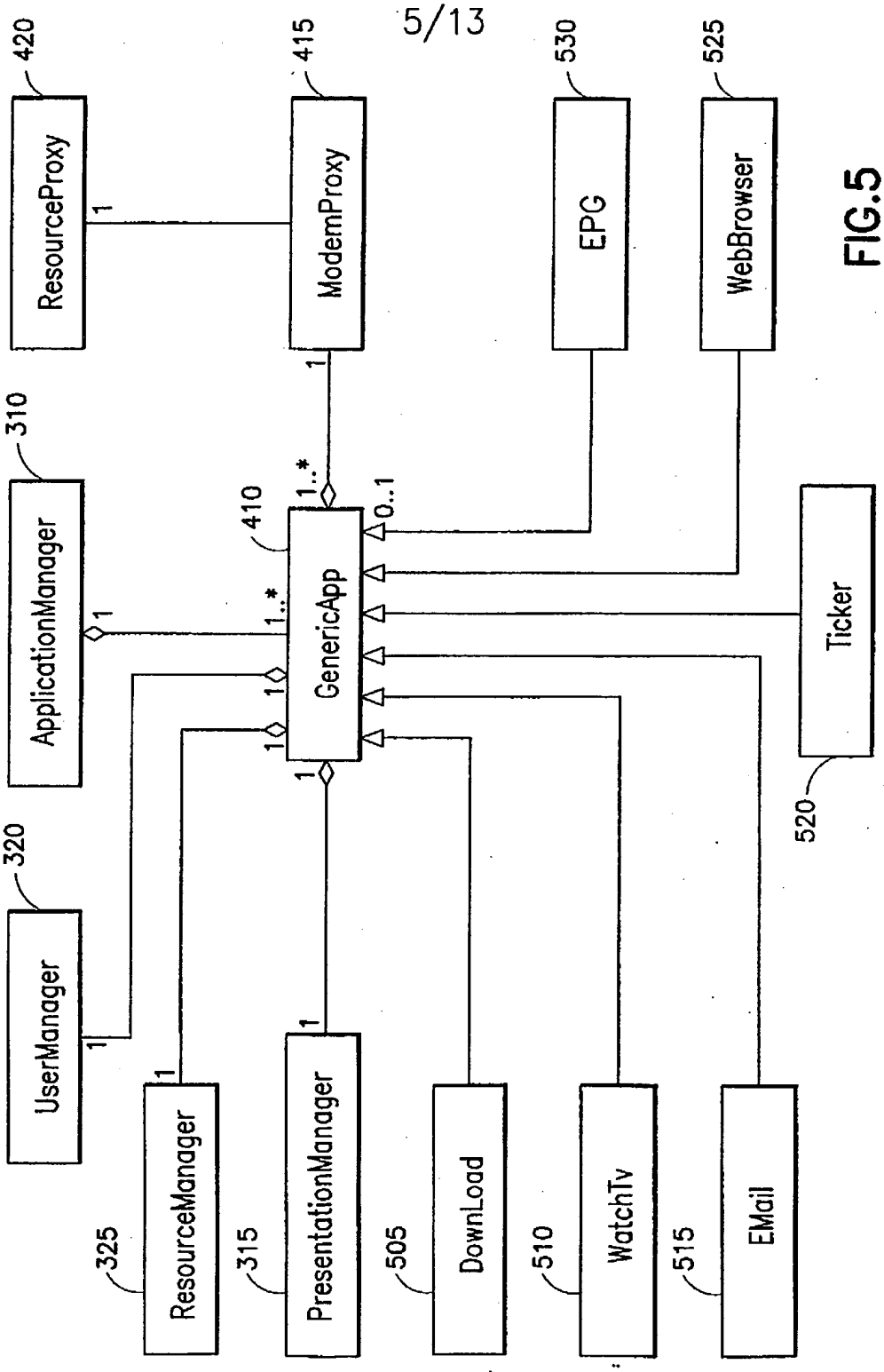

420
ResourceProxy
415
ModemProxy
530
EPG
525
WebBrowser
310
ApplicationManager
410
GenericApp
Ticker
520
320
UserManager
325
ResourceManager
315
PresentationManager
505
DownLoad
510
WatchTv
515
EMail
1
1..*
0..1

FIG.5

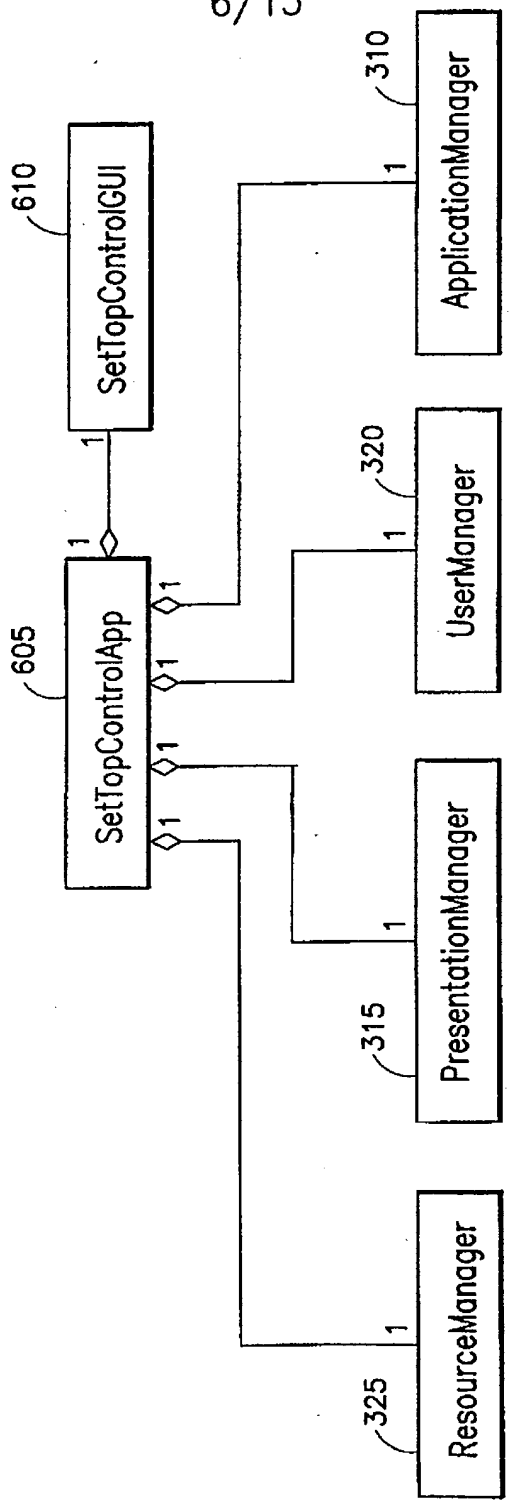

610
SetTopControlGUI
605
SetTopControlApp
310
ApplicationManager
320
UserManager
315
PresentationManager
325
ResourceManager
1
1
1
1
1
1
1
1
1
1

FIG.6

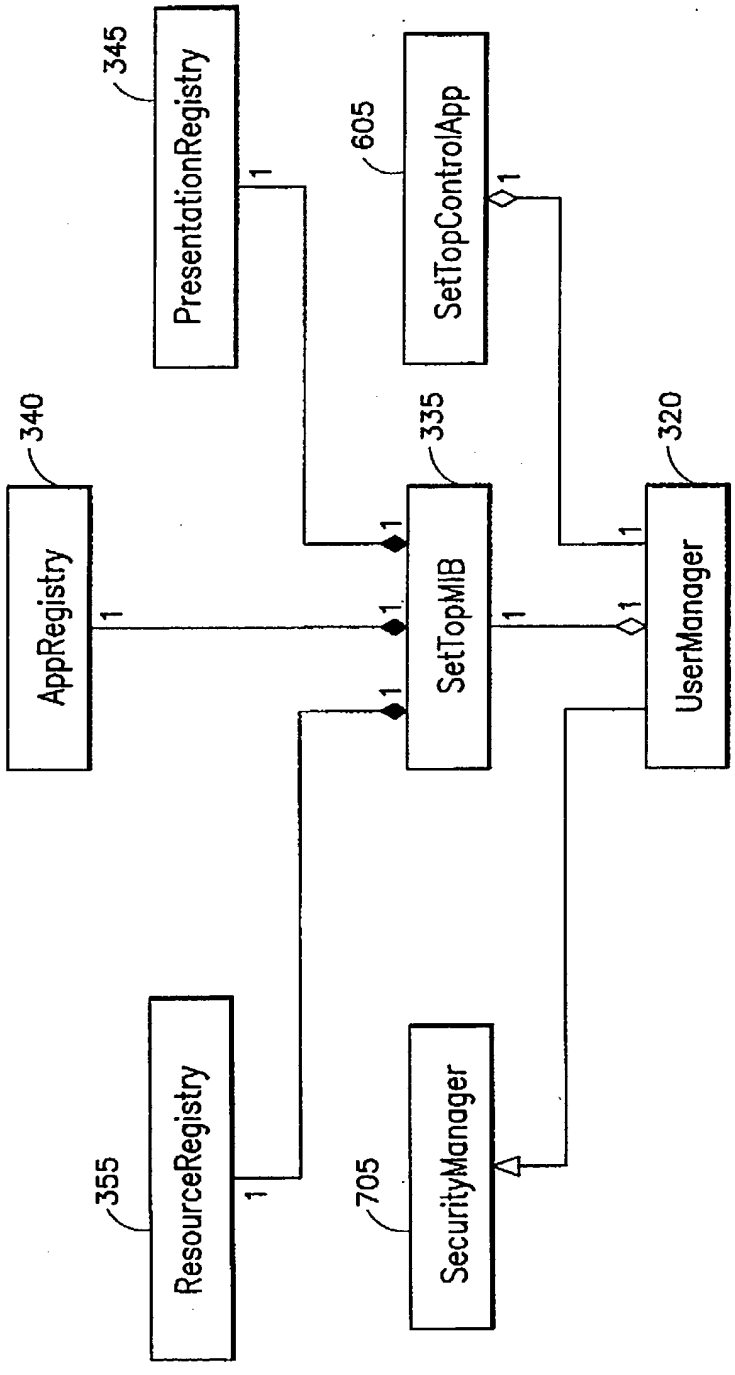

345
PresentationRegistry
605
SetTopControlApp
340
AppRegistry
335
SetTopMIB
320
UserManager
355
ResourceRegistry
705
SecurityManager
1

FIG.7

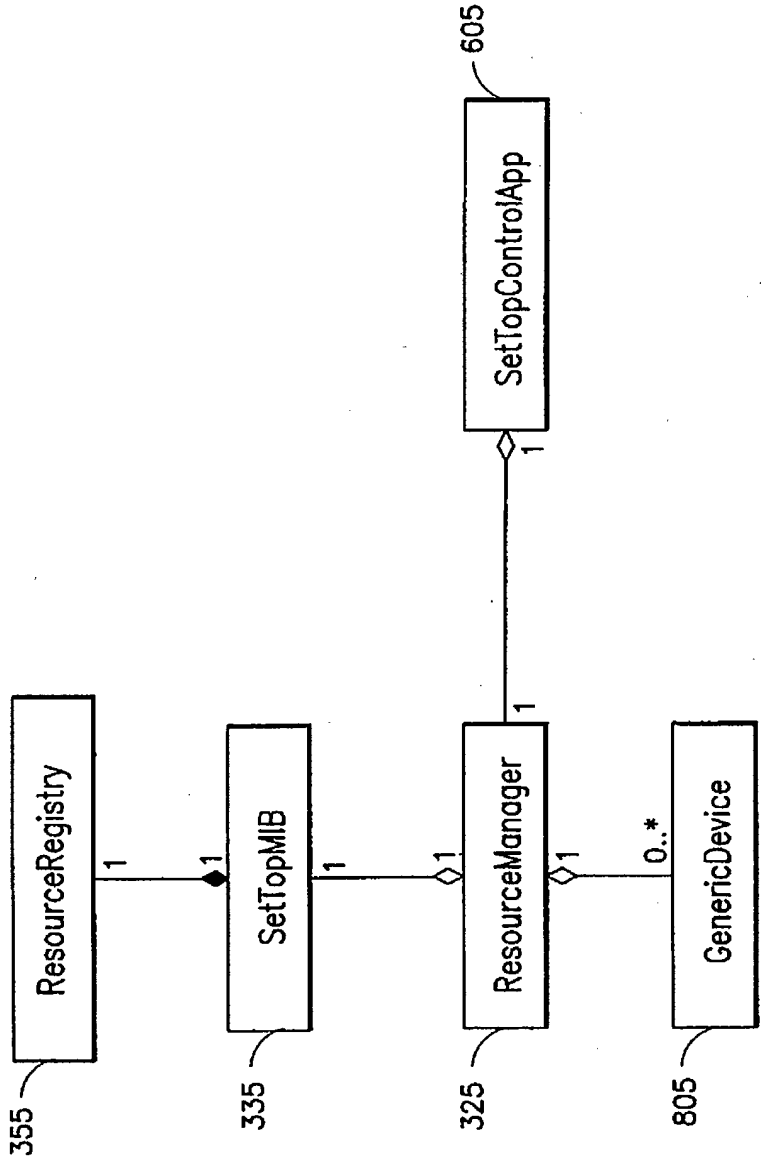

605
SetTopControlApp
1
1
ResourceRegistry
SetTopMIB
ResourceManager
GenericDevice
1
1
1
1
1
1
0..*
355
335
325
805

FIG.8

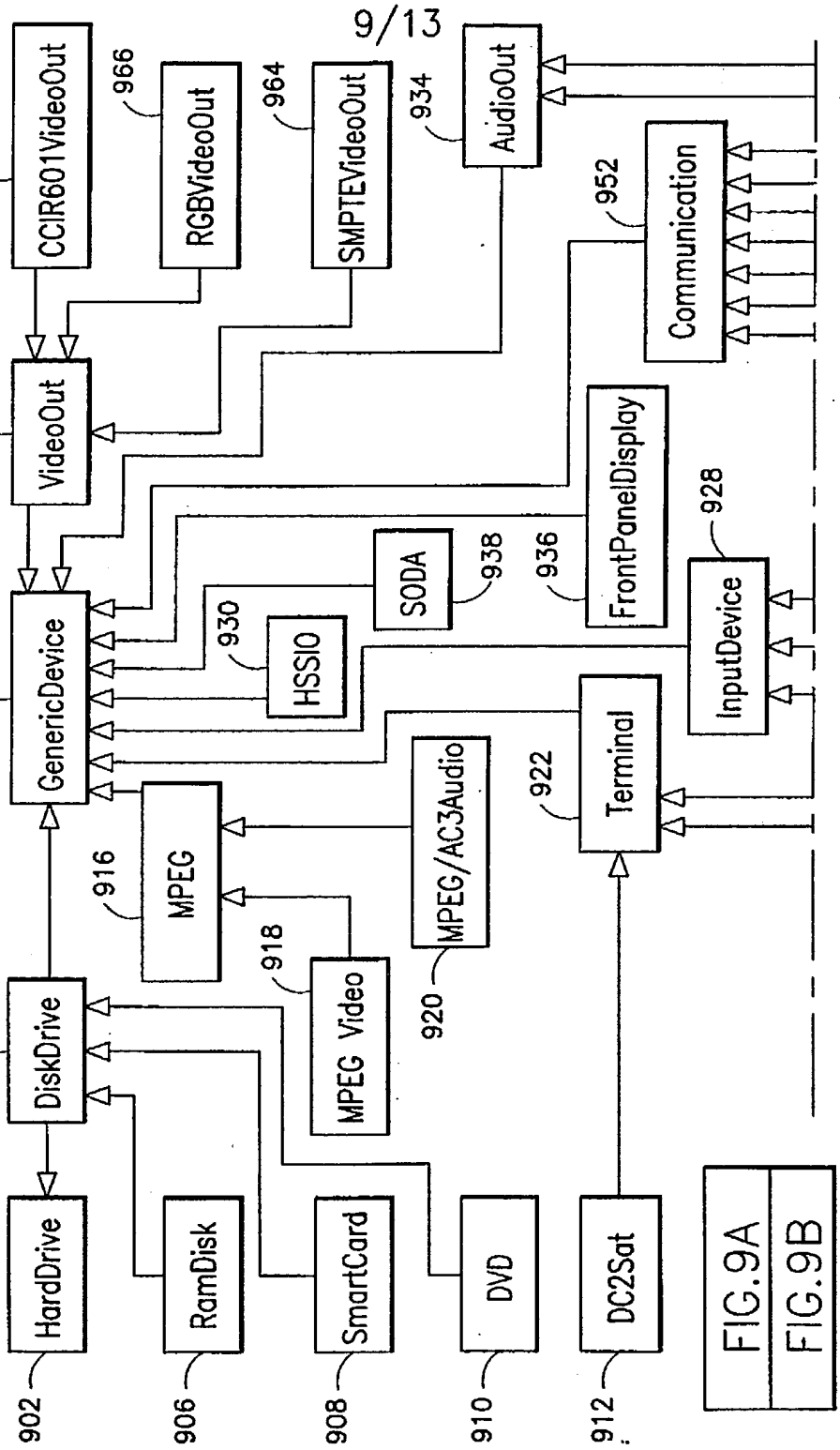

968
CCIR601VideoOut
966
RGBVideoOut
964
SMPTEVideoOut
934
AudioOut
952
Communication
932
VideoOut
936
FrontPanelDisplay
928
805
GenericDevice
930
HSSIO
938
SODA
InputDevice
922
Terminal
916
MPEG
MPEG/AC3Audio
918
MPEG Video
920
904
DiskDrive
902
HardDrive
906
RamDisk
908
SmartCard
910
DVD
912
DC2Sat
FIG.9A
FIG.9B

FIG.9A

FIG.9

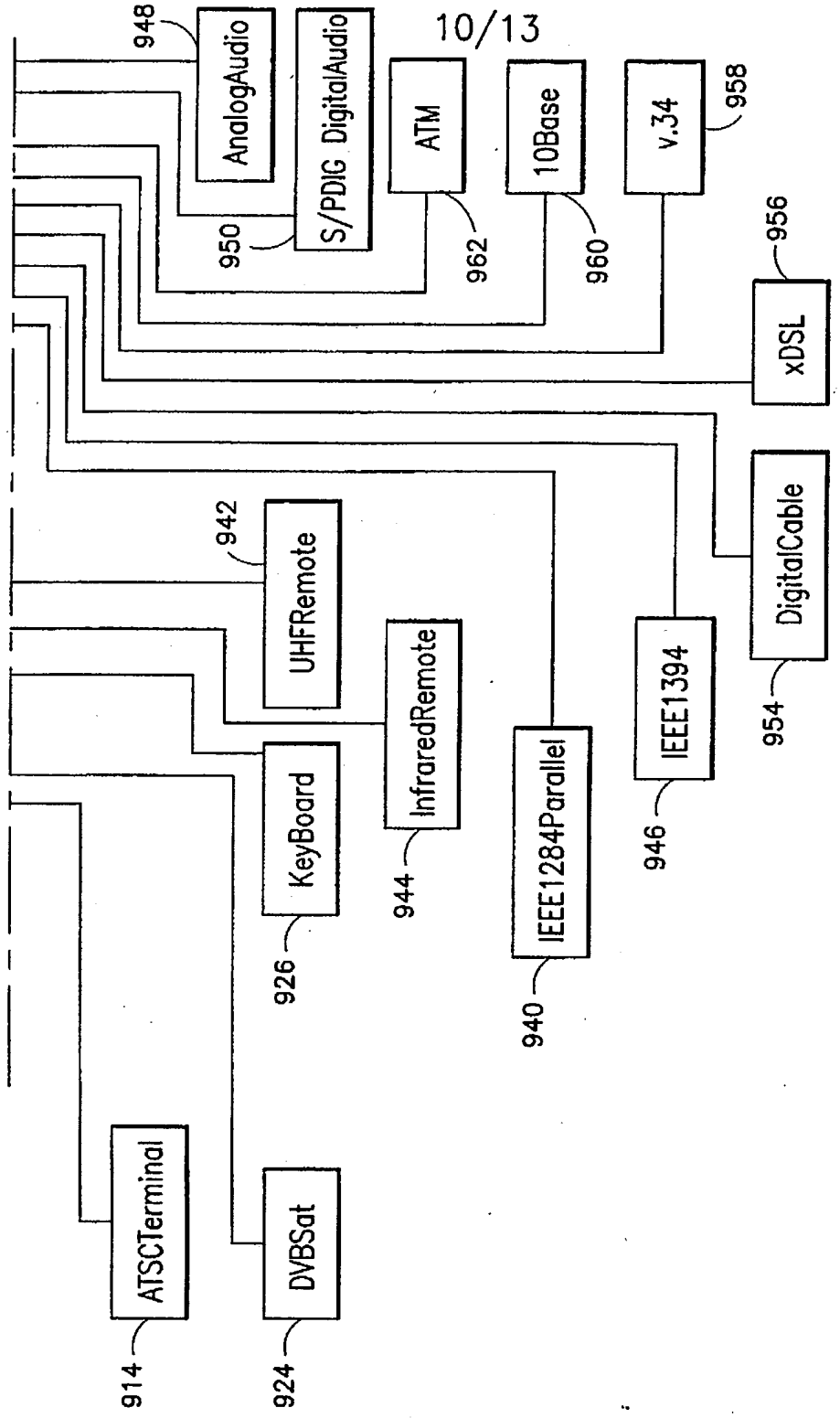

948
AnalogAudio
950
S/PDIF DigitalAudio
962
ATM
960
10Base
958
v.34
956
xDSL
954
DigitalCable
942
UHFRemote
944
InfraredRemote
926
KeyBoard
940
IEEE1284Parallel
946
IEEE1394
914
ATSCTerminal
924
DVBSat

FIG.9B

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter: ...al Application No
PCT/US 99/21983

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0908821 A	14-04-1999	AU 9363298 A	27-04-1999
		WO 9918730 A	15-04-1999
EP 0852443 A	08-07-1998	JP 10271484 A	09-10-1998
EP 0869447 A	07-10-1998	JP 10312432 A	24-11-1998
EP 0813147 A	17-12-1997	US 5946487 A	31-08-1999
		JP 10069394 A	10-03-1998

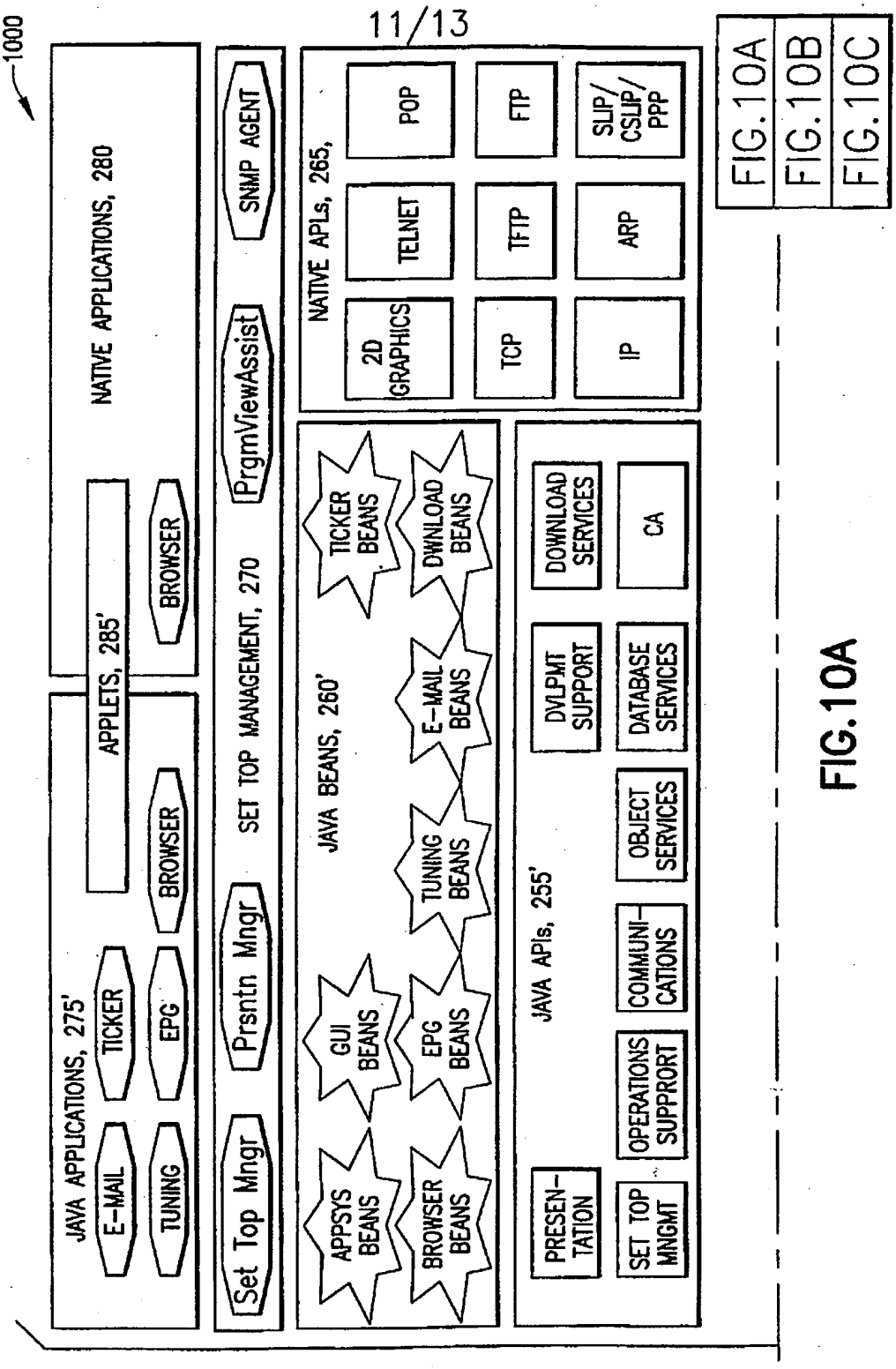


FIG.10A

FIG.10A
FIG.10B
FIG.10C

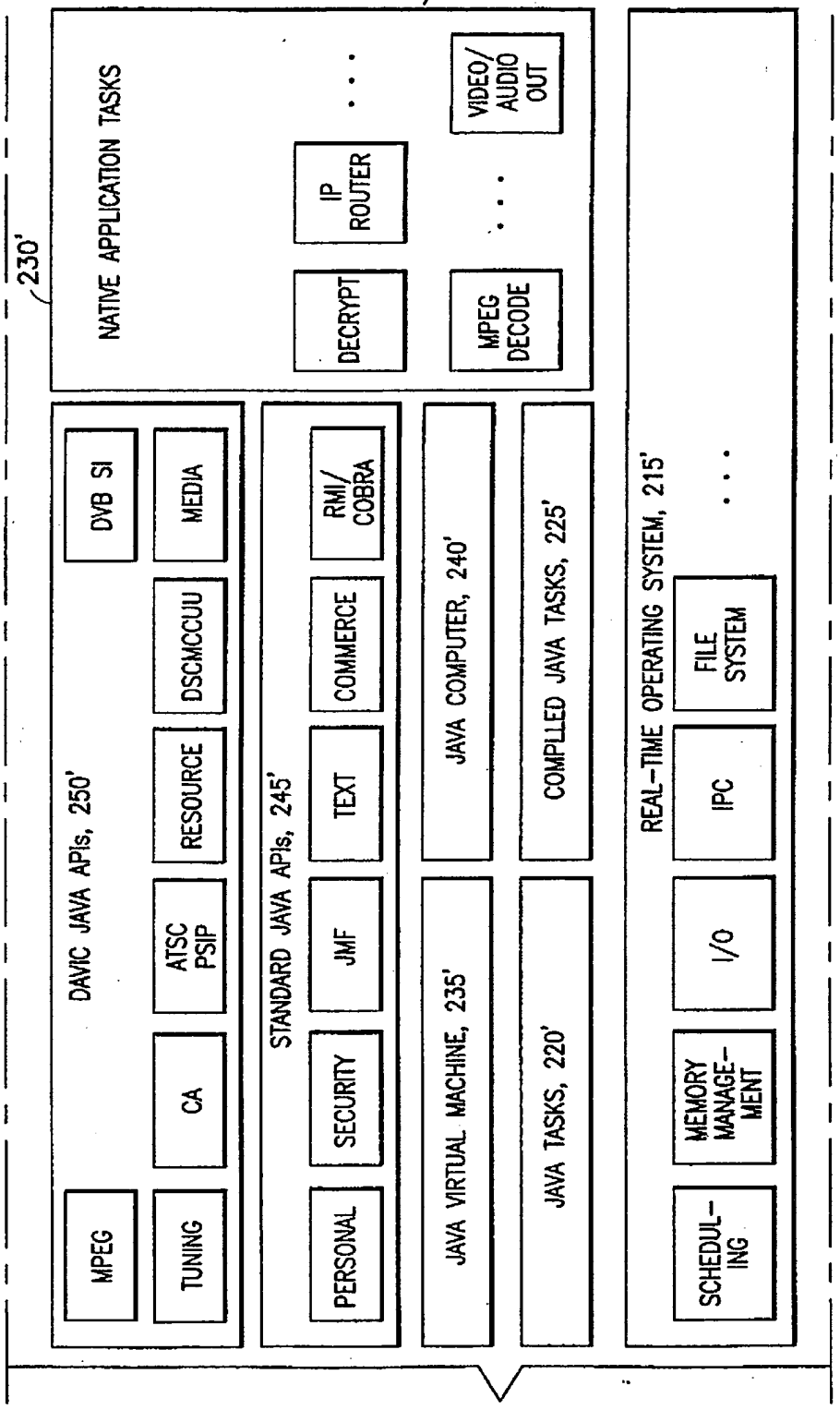

230'
NATIVE APPLICATION TASKS
DECRYPT
IP ROUTER
MPEG DECODE
VIDEO/ AUDIO OUT
DAVIC JAVA APIs, 250'
MPEG
TUNING
CA
ATSC PSIP
RESOURCE
DSCMCCUU
DVB SI
MEDIA
STANDARD JAVA APIs, 245'
PERSONAL
SECURITY
JMF
TEXT
COMMERCE
RMI/ COBRA
JAVA VIRTUAL MACHINE, 235'
JAVA COMPUTER, 240'
JAVA TASKS, 220'
COMPILED JAVA TASKS, 225'
REAL-TIME OPERATING SYSTEM, 215'
SCHEDUL- ING
MEMORY MANAGE- MENT
I/O
IPC
FILE SYSTEM

FIG.10B

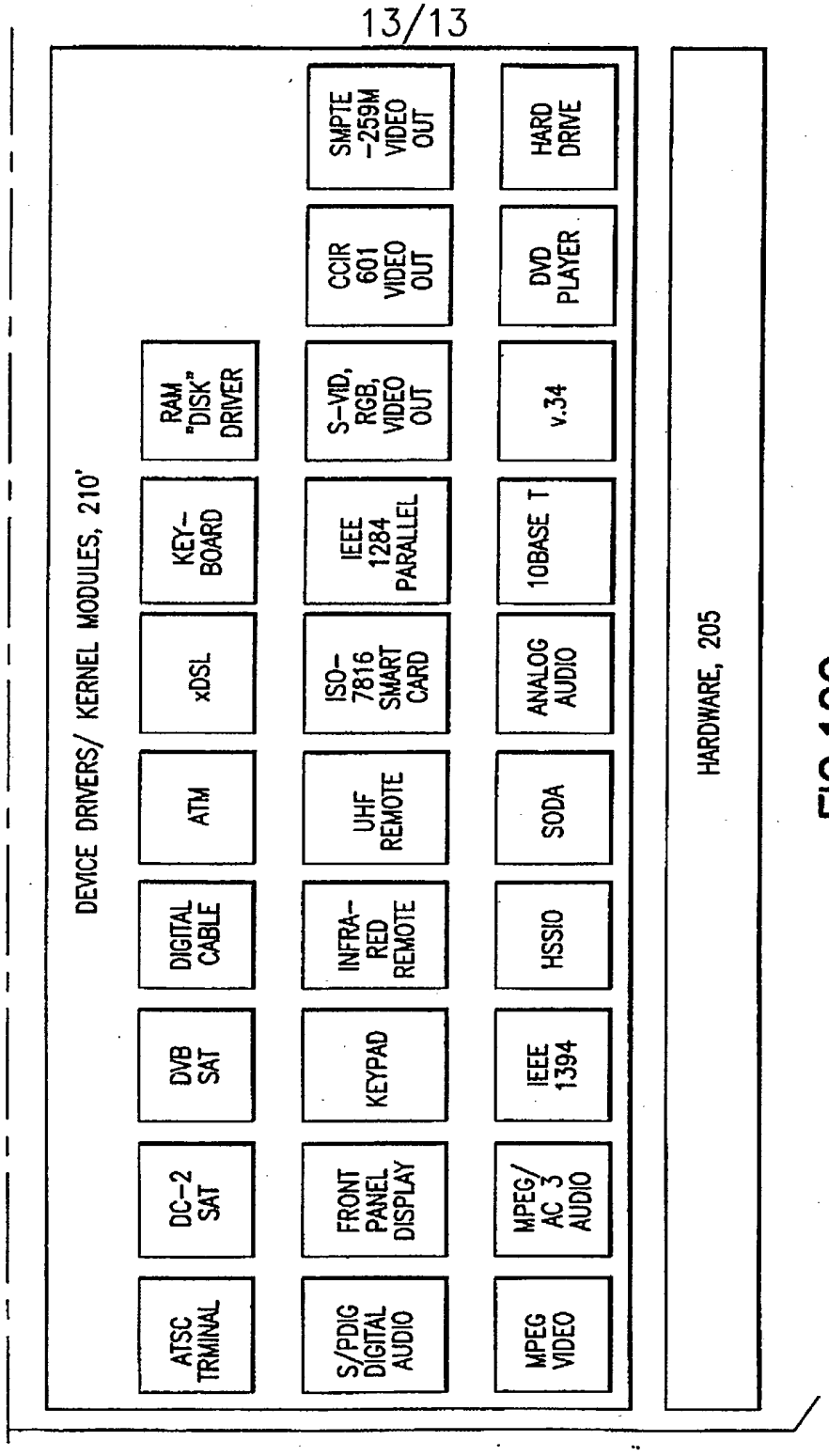

DEVICE DRIVERS/ KERNEL MODULES, 210'
ATSC TRMINAL
DC-2 SAT
DVB SAT
DIGITAL CABLE
ATM
xDSL
KEY-BOARD
RAM "DISK" DRIVER
S/PDIG DIGITAL AUDIO
FRONT PANEL DISPLAY
KEYPAD
INFRA-RED REMOTE
UHF REMOTE
ISO-7816 SMART CARD
IEEE 1284 PARALLEL
S-VID, RGB, VIDEO OUT
CCIR 601 VIDEO OUT
SMPTE -259M VIDEO OUT
MPEG VIDEO
MPEG/ AC 3 AUDIO
IEEE 1394
HSSIO
SODA
ANALOG AUDIO
10BASE T
v.34
DVD PLAYER
HARD DRIVE
HARDWARE, 205

FIG.10C

INTERNATIONAL SEARCH REPORT

Inter ıal Application No
PCT/US 99/21983

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N5/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	EP 0 908 821 A (CANAL PLUS SA) 14 April 1999 (1999-04-14) abstract; figure 3 ---	1,12
X	EVAIN J -P: "THE MULTIMEDIA HOME PLATFORM" EBU REVIEW- TECHNICAL,BE,EUROPEAN BROADCASTING UNION. BRUSSELS, no. 275, - 21 March 1998 (1998-03-21) page 4-10 XP000767493 ISSN: 0251-0936 the whole document --- -/--	1-16

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 January 2000	21/01/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Giannotti, P

3

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter ·al Application No
PCT/US 99/21983

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WILLIAMS T: "STB OPERATING SYSTEMS GEAR UP FOR FLOOD OF DATA SERVICES" COMPUTER DESIGN,US,PENNWELL PUBL. LITTLETON, MASSACHUSETTS, vol. 35, no. 2, - February 1996 (1996-02) page 67-68,72,74-76 XP000555483 ISSN: 0010-4566 the whole document	1-16
A	RATH K ET AL: "SET-TOP BOX CONTROL SOFTWARE: A KEY COMPONENT IN DITITAL VIDEO" PHILIPS JOURNAL OF RESEARCH,NL,ELSEVIER, AMSTERDAM, vol. 50, no. 1/02, - July 1996 (1996-07) page 185-199 XP000627669 ISSN: 0165-5817 abstract; figure 3	1-16
A	ITOH J -I ET AL: "A NEW SOFTWARE ARCHITECTURE FOR EVOLVABLE MULTIMEDIA SOFTWARE" EUROPEAN TRANSACTIONS ON TELECOMMUNICATIONS,IT,EUREL PUBLICATION, MILANO, vol. 8, no. 4, - July 1997 (1997-07) page 423-435 XP000695449 ISSN: 1124-318X abstract; figure 1	1-16
A	EP 0 852 443 A (TEXAS INSTRUMENTS INC) 8 July 1998 (1998-07-08) abstract; figure 2 column 4, line 54 -column 5, line 7	1,12
A	EP 0 869 447 A (COMPAQ COMPUTER CORP) 7 October 1998 (1998-10-07) abstract; figure 2 page 4, line 24 -page 5, line 13	1,12
A	EP 0 813 147 A (LSI LOGIC CORP) 17 December 1997 (1997-12-17)	

3

Form PCT/ISA/210 (continuation of second sheet) (July 1992)